Sotiris Nikoletseas
José D. P. Rolim (Eds.)

Theoretical Aspects of Distributed Computing in Sensor Networks

Springer

Sotiris Nikoletseas · José D.P. Rolim
Editors

Theoretical Aspects of Distributed Computing in Sensor Networks

Springer

Editors

Prof. Sotiris Nikoletseas
Dept. of Computer Engineering and Informatics
University of Patras
GR 265 00 Patras
Greece
and
Computer Technology Institute
N. Kazantzaki Str. 1
Patras University Campus
26504 Rion, Patras
Greece
nikole@cti.gr

Prof. José D.P. Rolim
Université de Genève
Centre Universitaire d'Informatique
Battelle Bat. A
7 rte de Drize
CH-1227 Carouge
Switzerland
jose.rolim@unige.ch

ISSN 1431-2654
ISBN 978-3-642-14848-4 e-ISBN 978-3-642-14849-1
DOI 10.1007/978-3-642-14849-1
Springer Heidelberg Dordrecht London New York

ACM Computing Classification (1998): C.2, F.2, G.2, E.4, E.1

Cover design: KuenkelLopka GmbH, Heidelberg

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Foreword

We first thank all authors of this book for their high-quality contributions; the willingness of these leading researchers to participate in this effort has been vital to its success and a great honor for us.

This book is organized into eight thematic sections, namely (i) Challenges for Wireless Sensor Networks; (ii) Models, Topology, Connectivity; (iii) Localization, Time Synchronization, Coordination; (iv) Data Propagation and Collection; (v) Energy Optimization; (vi) Mobility Management; (vii) Security Aspects; and (viii) Tools, Applications, and Use Cases. It spans a large spectrum of fundamental aspects for sensor networks such as topology management, node interactions and connectivity, information dissemination and obstacle avoidance, localization and synchronization, mobility management and robotic entities, energy optimization and power assignment, security, as well as relations of sensor networks to other network types such as radio and mobile. The book is meant for use by researchers, developers, educators, and students interested in the area of sensor networks. Since sensor networking is applied to a variety of domains, the ideas, methods, and techniques illustrated in this book may be useful to a very wide audience. Furthermore, this book can also be used as a supplement to any course on algorithms, wireless protocols, and distributed computing and networking.

We wish to thank Springer, Lecture Notes in Computer Science (LNCS), for publishing this book; in particular, we thank Alfred Hofmann for our long cooperation and his persistent willingness to encourage high-quality publications in emerging research topics. Also, we thank his Springer colleague Ronan Nugent for a fruitful cooperation in realizing this volume. Many thanks go to Dionysios Efstathiou (MSc), a brilliant PhD student at the University of Patras and CTI, actively working on sensor networks, for integrating the Volume material so timely and efficiently.

Finally, we acknowledge support from the EU research project AEOLUS ("Algorithmic Principles for Building Efficient Overlay Computers") of the Sixth Framework Programme/FET Proactive Initiative on Global Computing. Several of our research perspectives have been positively influenced by AEOLUS, while many AEOLUS researchers are contributors of this book.

Patras, Greece — Sotiris Nikoletseas
Geneva, Switzerland — José D.P. Rolim
November 2010

Preface

Wireless ad hoc sensor networks have recently become a very active research subject as well as a topic of rapid technological progress and large-scale practical development and application activities. However, a solid foundational and theoretical background seems still necessary for sensor networking to achieve its full potential. The provision of relevant abstract modeling, novel algorithmic design, and analysis methodologies toward efficient and robust realizations of such very large, highly dynamic, complex, non-conventional networks is a challenging task for the theoretical computer science community (and that of distributed computing in particular).

Several models, algorithms, and interesting research results have already appeared, in specialized and generic theory journals, conferences and workshops. This book aims to reinforce the emergence of a critical mass of theoretical and algorithmic foundations by bringing together, for the first time in a systematic way, high-quality research contributions (invited book chapters) by leading experts worldwide relevant to important algorithmic and complexity-theoretic aspects of wireless sensor networks and related ad hoc network types.

The intended audience of this book includes researchers and advanced graduate students working on sensor networks and the broader areas of wireless networking and distributed computing, as well as practitioners in the relevant application domains interested to obtain a broader foundational perspective and insight. The book can also serve as a text for advanced university courses and research seminars.

The book is structured into eight themes covering respective common aspects, issues, and methodologies. This division is rather indicative; because of the inherent relations of different topics, layers, and problems, many chapters could be associated to more than one theme, and the themes themselves could have been chosen in a different manner. Still, we hope that the particular structure will be methodologically useful for the reader.

We now briefly describe each theme. The first one discusses characteristic challenges for distributed sensor networking; although the perspective stems from systems methodologies, the implications to algorithms and theory are relevant.

The second theme presents current abstract modeling proposals for sensor networks related to different layers (physical, network), diverse (yet highly related) aspects such as the topology management and mobility plane, as well as the important aspect of network coding.

The next theme concerns basic primitives for distributed computing in sensor networks such as localization, time synchronization, and decentralized coordination. Efficient distributed solutions to such primitives are necessary for higher layer network services, such as the (rather canonical) problem of data routing. Data routing (and information dissemination, more general) is studied in the fourth theme, in terms of both propagating data to a sink destination as well as collecting data from the network nodes.

The fifth theme covers one of the most important challenges in sensor networking, that of energy optimization. Different aspects of energy management are addressed, such as prolonging the network lifetime via probabilistically optimized routing decisions as well as via mobility optimization. This mobility-based approach nicely connects to the next theme which addresses mobility and its complications; also, how to exploit mobility anyway present in the network to, e.g., optimize information spreading.

The important aspect of security in sensor networks is investigated in the seventh theme by addressing complementary aspects, such as the efficient distribution and management of secure keys. The book concludes with an interesting more practical theme on characteristic applications and representative tools for programming sensor networks, as well as the discussion of a use case scenario in the context of a Future Internet perspective.

We hope that this book will be helpful to its readers and contribute to a solid foundation and deeper understanding of the fascinating and rapidly evolving research area of distributed sensor networking.

Patras, Greece — Sotiris Nikoletseas
Geneva, Switzerland — José D.P. Rolim
November 2010

Contents

Contributors

Tarek Abdelzaher University of Illinois at Urbana-Champaign, Champaign, IL, USA, zaher@cs.uiuc.edu

Luzi Anderegg Institute of Theoretical Computer Science, ETH Zürich, Zurich, Switzerland, anderegg@inf.ethz.ch

Przemysław Błaśkiewicz Wrocław University of Technology, Wrocław, Poland, Przemyslaw.Blaskiewicz@pwr.wroc.pl

Azzedine Boukerche University of Ottawa, Ottawa, ON, Canada, boukerch@site.uottawa.ca

Costas Busch Department of Computer Science, Louisiana State University, Baton Rouge, LA, USA, busch@csc.lsu.edu

Mung Chiang Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, chiangm@princeton.edu

Andrea Clementi Dipartimento di Matematica, Università di Roma "Tor Vergata", Roma, Italy, clementi@mat.uniroma2.it

Sajal K. Das Department of Computer Science and Engineering, Center for Research in Wireless Mobility and Networking (CReWMaN), University of Texas at Arlington, Arlington, TX, USA, das@uta.edu

Josep Díaz Departament de Llenguatges, Sistemes i Informàtics, Universitat Politècnica de Catalunya, Grup de Recerca ALBCOM, Barcelona, Spain diaz@lsi.upc.edu

Shlomi Dolev Department of Computer Science, Ben-Gurion University of the Negev, Beer-sheva, Israel, dolev@cs.bgu.ac.il

Ozlem Durmaz Incel NETLAB, Department of Computer Engineering, Bogazici University, Istanbul, Turkey, ozlem.durmaz@tam.boun.edu.tr

Jeffrey Dwoskin Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, jdwoskin@princeton.edu

Stephan Eidenbenz Information Sciences, Los Alamos National Laboratory, Los Alamos, NM, USA, eidenben@lanl.gov

Alexander Fanghänel RWTH Aachen University, Aachen, Germany, fanghaenel@cs.rwth-aachen.de

Paola Flocchini School of Information Technology and Engineering, University of Ottawa, Ottawa, ON, Canada, flocchin@site.uottawa.ca

Amitabha Ghosh Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA, amitabhg@usc.edu

R. K. Ghosh Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, India, rkg@cse.iitk.ac.in

Arvind Giridhar Goldman Sachs, New York, NY, USA, arvind.giridhar@gmail.com

Tian He Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA, tianhe@cs.umn.edu

Jianwei Huang Department of Information Engineering, The Chinese University of Hong Kong, New Territories, Hong Kong, jwhuang@ie.cuhk.edu.hk

Dariusz R. Kowalski Department of Computer Science, University of Liverpool, Liverpool, UK, D.Kowalski@liverpool.ac.uk

Evangelos Kranakis School of Computer Science, Carleton University, Ottawa, ON, Canada, kranakis@scs.carleton.ca

Srdjan Krco Ericsson, Belgrade, Serbia, srdjan.krco@ericsson.com

Bhaskar Krishnamachari Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA, bkrishna@usc.edu

Danny Krizanc Department of Mathematics and Computer Science, Wesleyan University, Middletown, CT, USA, dkrizanc@wesleyan.edu

P.R. Kumar CSL and Department of ECE, University of Illinois, Urbana, IL, USA, prkumar@illinois.edu

Mirosław Kutyłowski Wrocław University of Technology, Wrocław, Poland, Miroslaw.Kutylowski@pwr.wroc.pl

Tian Lan Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, tlan@princeton.edu

Ruby Lee Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, rblee@princeton.edu

Pierre Leone University of Geneva, Geneva, Switzerland, pierre.leone@unige.ch

Shizheng Li Iowa State University, Ames, IA, USA, szli@iastate.edu

Jun Luo School of Computer Engineering, Nanyang Technological University (NTU), Nayang Avenue, Singapore, junluo@ntu.edu.sg

Malik Magdon-Ismail Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA, magdon@cs.rpi.edu

Anahit Martirosyan University of Ottawa, Ottawa, ON, Canada, amart013@uottawa.ca

Dieter Mitsche Universitat Politecnica de Catalunya Grup de Recerca ALBCOM, Barcelona, Spain, dmitche@gmail.com

Nathalie Mitton INRIA Lille - Nord Europe/University of Lille, Lille, France, nathalie.mitton@inria.fr

Oscar Morales School of Computer Science, Carleton University, Ottawa, ON, Canada, oscarmponce@gmail.com

Alfredo Navarra Department of Mathematics and Computer Science, University of Perugia, Perugia, Italy, navarra@dmi.unipg.it

Amiya Nayak SITE, University of Ottawa, Ottawa, ON, Canada, anayak@site.uottawa.ca

Sotiris Nikoletseas Research Academic Computer Technology Institute (RACTI) and Department of Computer Engineering and Informatics, University of Patras, Patras, Greece, nikole@cti.gr

Francesco Pasquale Dipartimento di Informatica e Applicazioni “R.M. Capocelli”, Università di Salerno, Salerno, Italy, pasquale@dia.unisa.it

Animesh Pathak INRIA Paris-Rocquencourt, Rocquencourt, France, animesh.pathak@inria.fr

Leon Peeters Institute of Theoretical Computer Science, ETH Zürich, Zurich, Switzerland, peeters@inf.ethz.ch

Cristina M. Pinotti Department of Mathematics and Computer Science, University of Perugia, Perugia, Italy, pinotti@unipg.it

Viktor K. Prasanna University of Southern California, Los Angeles, CA, USA, prasanna@usc.edu

Giuseppe Prencipe Dipartimento di Informatica, University of Pisa, Pisa, Italy, prencipe@di.unipi.it

Aditya Ramamoorthy Iowa State University, Ames, IA, USA, adityar@iastate.edu

Tahiry Razafindralambo INRIA Lille - Nord Europe / University of Lille, Lille, France, tahiry.razafindralambo@univ-lille1.fr

Rupert Reiger EADS, Innovation Works, Munich, Germany, Rupert.Reiger@eads.net

José D.P. Rolim University of Geneva, Geneva, Switzerland, Jose.Rolim@unige.ch

Sushmita Ruj Department of Electrical and Information Technology, Lund University, Lund, Scania, Sweden, Sushmita.Ruj@eit.lth.se

Paolo Santi Istituto di Informatica e Telematica del CNR, Pisa, Italy, paolo.santi@iit.cnr.it

Nicola Santoro School of Computer Science, Carleton University, Ottawa, ON, Canada, santoro@scs.carleton.ca

David Simplot-Ryl INRIA Lille - Nord Europe / University of Lille, Lille, France, Simplot-Ryl@inria.fr

Paul G. Spirakis Research Academic Computer Technology Institute (RACTI) and Department of Computer Engineering and Informatics, University of Patras, Patras, Greece, spirakis@cti.gr

John A. Stankovic Department of Computer Science, University of Virginia, Charlottesville, VA, USA, stankovic@cs.virginia.edu

Ivan Stojmenovic SITE, University of Ottawa, Ottawa, ON, Canada, ivan@site.uottawa.ca

Nir Tzachar Department of Computer Science, Ben-Gurion University of the Negev, Beer-sheva, Israel, tzachar@cs.bgu.ac.il

Berthold Vöcking RWTH Aachen University, Aachen, NRW, Germany, voecking@cs.rwth-aachen.de

Jing Wang Department of Computer Science and Engineering, Center for Research in Wireless Mobility and Networking (CReWMaN), University of Texas at Arlington, Arlington, TX USA, jingwang@uta.edu

Peter Widmayer Institute of Theoretical Computer Science, ETH Zürich, Zürich, Switzerland, widmayer@inf.ethz.ch

Anthony D. Wood Department of Computer Science, University of Virginia, Charlottesville, VA, USA, wood@cs.virginia.edu

Konrad Wrona NATO C3 Agency, The Hague, Netherlands, konrad.wrona@nc3a.nato.int

Jing Xi Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA, xij2@cs.rpi.edu

Liu Xiang School of Computer Engineering, Nanyang Technological University (NTU), Nanyang Avenue, Singapore, xiangliu@pmail.ntu.edu.sg

Dahai Xu AT&T Labs Research, Florham Park, NJ, USA, dahaixu@research.att.com

Part I
Challenges for Wireless Sensor Networks

Chapter 1
Composition and Scaling Challenges in Sensor Networks: An Interaction-Centric View

T. Abdelzaher

Abstract Moore's law, automation considerations, and the pervasive need for timely information lead to a next generation of distributed systems that are open, highly interconnected, and deeply embedded in the physical world by virtue of pervasive sensing and sensor-based decision-making. These systems offer new research challenges that stem from scale, composition of large numbers of components, and tight coupling between computation, communication, and distributed interaction with both physical and social contexts. These growing challenges span a large spectrum ranging from new models of computation for systems that live in physical and social spaces, to the enforcement of reliable, predictable, and timely end-to-end behavior in the face of high interactive complexity, increased uncertainty, and imperfect implementation. This chapter discusses the top challenges in composing large-scale sensing systems and conjectures on research directions of increasing interest in this realm.

1.1 Introduction

The envisioned proliferation of networked sensing devices, predicted in the 1990s, has given rise to myriads of challenges that arise from interconnecting sensors at large scale. Future distributed sensing systems will surpass the current paradigms for embedded computing, where a number of sensors and actuators implement well-understood and tightly managed control loops. New paradigms will involve data acquisition at a significantly wider scale, offering less structure and less control over the properties of the resulting loops from sensing to decision-making. It is envisioned that by the end of the next decade, a significant number (if not the majority) of Internet clients will constitute sensors and embedded devices. Indeed, the main role of future networks will shift from offering a mere communication medium between end-points to offering *information distillation* services bridging the gap between the

T. Abdelzaher (✉)
University of Illinois at Urbana-Champaign, Champaign, IL, USA
e-mail: zaher@cs.uiuc.edu

myriads of real-time low-level data feeds and the high-level human decision needs. The success of Google, built around the mission of organizing and “distilling” Web content, attests to the increasing use of networks as information sources. The proliferation of sensing devices gives rise to new information acquisition paradigms, such as opportunistic and participatory sensing [1, 4, 9, 15, 18], that rely on sensory data collection by individuals or devices acting on their behalf and on sharing these data at large scale to extract information of common use. New challenges arise in supporting the information distillation requirements of such emerging applications. These challenges are brought about primarily by scale and the need to compose sensing systems of large numbers of components, while maintaining predictable end-to-end properties.

Composition challenges arise from the complex interactions that large-scale sensing systems exhibit in several spaces including functional, data, and temporal interactions. This chapter focuses on four important interaction challenges that arise by virtue of scale. Namely, we elaborate on composition challenges in the face of functional interactions, data interactions, timing interactions, and interactions of system dynamics in largely distributed sensing systems.

It should be noted that this chapter is by no means a complete account of sensor network design and performance challenges. Most prominently, the chapter does not directly address the issues of heterogeneity, programming interfaces, middleware, and architectural paradigms used to facilitate building large systems. These software and architectural solutions, as well as examples of large-scale deployed networks, are detailed elsewhere in this book. The chapter also does not address the issues of security; a growing concern in recent literature as sensor networks empark on new mission-critical application domains where secure operation must be assured. The interaction challenges mentioned above present a more basic categorization of challenges, classifying them not by the software layer in which they arise (such as operating systems, communication protocols, middleware, or programming support), but rather by the conceptual space in which they occur, such as functional, temporal, or data related. These spaces are described in the following sections, respectively.

1.2 Functional Interactions

The first interaction space for components of sensor network applications is the space of *functional interactions*. Most deployed distributed system failures are attributed to unexpected interactions between multiple components that lead to new subtle failure modes. In sensor networks, the space of such interactions often cannot be fully explored at design time. Significant advances have been made in the area of formal methods and verification techniques. However, they remain limited by scalability challenges that arise due to massive concurrency and unreliable components (e.g., wireless). Most of the system development time, therefore, is spent on debugging as opposed to new component development. Debugging individual components is relatively simple. The problem lies in understanding failures that arise due to component composition in complex systems.

1.2.1 Troubleshooting Interactive Complexity

New fundamental theory, algorithms, analysis techniques, and software tools are needed to help uncover root causes of errors resulting from interactions of large numbers of components in heterogeneous networked sensing systems. System heterogeneity and tight integration between computation, communication, sensing, and control lead to a high interactive complexity. Moreover, the lack of layering and isolation, attributed to resource constraints, make it hard to apply the usual software engineering techniques aimed at reducing interactive complexity, thus further increasing the possibility of distributed errors and unexpected failures. On the other hand, software reuse is impaired by the customized nature of application code and deployment environments, making it harder to amortize debugging, troubleshooting, and tuning cost. Hence, while individual devices and subsystems may operate well in isolation, their composition may result in incompatibilities, anomalies, or failures that are very difficult to troubleshoot. At the same time, users of such systems (such as domain scientists) may not be experts on networking and system administration.

Automated techniques are needed for troubleshooting the system both at development time and after deployment in order to reduce production as well as ownership costs. Using such automated techniques, developers of future networked sensing systems should be able to significantly curtail debugging effort. Similarly, upon network deployment, service agents should be able to quickly diagnose and resolve problems in unique customer installations, hence reducing ownership cost. The aim is to answer developer or user questions such as "Why does this sensor network suffer unusually high service delays?", "Why is data throughput low despite availability of resources and service requests?", "Why does my time synchronization protocol fail to synchronize clocks when the localization protocol is run concurrently?", or "Why does this vehicle tracking system suffer increased false alarms when it is windy?[1]" Building efficient troubleshooting support to address the above questions offers significant research challenges brought about by the nature of interaction bugs, such as:

- *Non-reproducible behavior:* Interactions in networked sensing systems feature an increased level of concurrency and thus an increased level of non-determinism. In turn, non-determinism generates non-reproducible bugs that are hard to find using traditional debugging tools.
- *Non-local emergent behavior:* By definition, interaction bugs do not manifest themselves when components are tested in isolation. Current debugging tools are very good at finding bugs that can be traced to individual components. Interaction bugs manifest only at scale as a result of component composition. They result in emergent behavior that arises when a number of seemingly individually sound components are combined into a network, which makes them hard to find.

[1] In a previous deployment of a magnetometer-based wireless tracking system, wind resulted in antennae vibration which was caught by the magnetometers and interpreted as the passage of nearby ferrous objects (vehicles).

A successful approach to the automation of diagnosis of interactive complexity failures must rely on two main design principles aimed at *exploiting* concurrency, interactions, and non-determinism to *improve* the ability to diagnose problems in resource-constrained systems. These principles are as follows:

- *Exploiting non-reproducible behavior:* Exploitation of non-determinism to improve understanding of system behavior is not new to computing literature. For example, many techniques in estimation theory, concerned with estimation of system models, rely on introducing noise to explore a wider range of system states and hence arrive at more accurate models. Machine learning and data mining approaches have the same desirable property. They require examples of both good and bad system behavior to be able to classify the conditions correlated with good and bad. In particular, note that conditions that cause a problem to occur are correlated (by causality) with the resulting bad behavior. Root causes of non-reproducible bugs are thus inherently suited for discovery using data mining and machine learning approaches as the lack of reproducibility itself and the inherent system non-determinism improve the odds of occurrence of sufficiently diverse behavior examples to train the troubleshooting system to understand the relevant correlations and identify causes of problems.
- *Exploiting interactive complexity:* Interactive complexity describes a system where scale and complexity cause components to interact in unexpected ways. A failure that occurs due to such unexpected interactions is therefore not localized and is hard to "blame" on any single component. This fundamentally changes the objective of a troubleshooting tool from aiding in stepping through code (which is more suitable for finding a localized error in some line, such as an incorrect pointer reference), to aiding with diagnosing a *sequence of events* (component interactions) that lead to a failure state. For example, sequence mining algorithms present a suitable core analytic engine for diagnostic debugging.

1.2.2 Troubleshooting Examples

An example application of the above principles is reported in a previous investigation of a diagnostic debugging tool prototype. This prototype was experimented with over the course of 1 year to understand the strengths and limitations of the aforementioned approach [24–26]. The examples below give a feel for how diagnostic debugging is used according to this investigation.

1.2.2.1 A "Design" Bug

As an example of catching a design bug, we summarize a case study, published in [26], involving a multi-channel sensor network MAC-layer protocol from prior literature [27] that attempts to utilize channel diversity to improve throughput. The protocol assigned a home channel to every node, grouping nodes that communicated much into a cluster on the same channel. It allowed occasional communication between nodes in different clusters by letting senders change their channel temporary to the home channel of a receiver to send a message. If communication failed (e.g., because home channel information became stale), senders would scan all channels looking for the receiver on a new channel and update home channel information accordingly. Testing revealed that total network throughput was sometimes worse than that of a single-channel MAC. Initially, the designer attributed it to the heavy cost of communication *across* clusters. To verify this hypothesis, the original protocol, written for MicaZ motes, was instrumented to log radio channel change events and message communication events (send, receive, acknowledge) as well as related timeouts. It was tested on a motes network. Event logs from runs where it outperformed a single-channel MAC were marked "good." Event logs from runs where it did worse were marked "bad." Discriminative sequence mining applied to the two sets of logs revealed a common pattern associated prominently with bad logs. The pattern included the events `No Ack Received`, `Retry Transmission on Channel (1)`, `Retry Transmission on Channel (2)`, `Retry Transmission on Channel (3)`, executed on a large number of nodes. This quickly led the designer to understand a much deeper problem. When a sender failed to communicate with a receiver in another cluster, it would leave its home channel and start scanning other channels causing communication addressed to it from upstream nodes in its cluster to fail as well. Those nodes would start scanning too, resulting in a cascading effect that propagated up the network until everyone was scanning and communication was entirely disrupted everywhere (both *within* and *across* clusters). The protocol had to be redesigned.

1.2.2.2 An "Accumulative Effect" Bug

Often failures or performance problems arise because of accumulative effects such as gradual memory leakage or clock overflow. While such effects evolve over a large period of time, the example summarized below [24] shows how it may be possible to use diagnostic debugging to understand the "tipping point" that causes the problem to manifest. In this case, the operators observed sudden onset of successive message loss in an implementation of directed diffusion [19], a well-known sensor network routing protocol. As before, communication was logged together with timeout and message drop events. Parts of logs coinciding with or closely preceding instances of successive message losses were labeled "bad." The rest were labeled "good." Discriminative sequence mining revealed the following sequential patterns correlated with successive message loss:

```
Message Send (timestamp = 255),    Message Send (timestamp = 0),
Message Dropped (Reason = "SameDataOlderTimeStamp").
```

The problem became apparent. A timestamp counter overflow caused subsequent messages received to be erroneously interpreted as "old" duplicates (i.e., having previously seen sequence numbers) and discarded.

1.2.2.3 A "Race Condition" Bug

Race conditions are a significant cause of failures in systems with a high degree of concurrency. A previous case study [26] demonstrated how diagnostic debugging helped catch a bug caused by a race condition in their embedded operating system called LiteOS [5]. When the communication subsystem of an early version of LiteOS was stress-tested, some nodes would crash occasionally and non-deterministically. LiteOS allows logging system call events. Such logs were collected from (the flash memory of) nodes that crashed and nodes that did not, giving rise to "good" and "bad" data sets, respectively. Discriminative sequence mining revealed that the following sequence of events occurred in the good logs but not in the bad logs: `Packet Received`, `Get Current Radio Handle`, whereas the following occurred in the bad logs but not in the good logs: `Packet Received`, `Get Serial Send Function`. From these observations, it is clear that failure occurs when `Packet Received` is followed by `Get Serial Send Function` before `Get Current Radio Handle` is called. Indeed, the latter prepares the application for receiving a new packet. At high data rates, another communication event may occur before the application is prepared, causing a node crash. The race condition was eliminated by proper synchronization.

1.2.2.4 An "Assumptions Mismatch" Bug

In systems that interact with the physical world, problems may occur when the assumptions made in protocol design regarding the physical environment do not match physical reality. In this case study [25], a distributed target tracking protocol, implemented on motes, occasionally generated spurious targets. The protocol required that nodes sensing a target form a group and elect a group leader who assigned a new ID to the target. Subsequently, the leader passed the target ID on in a leader handoff operation as the target moved. Communication logs were taken from runs where spurious targets were observed ("bad" logs) and runs where they were not ("good" logs). Discriminative pattern mining revealed the absence of member-to-leader messages in the bad logs. This suggested that a singleton group was present (i.e., the leader was the only group member). Indeed, it turned out that the leader hand off protocol was not designed to deal with a singleton group because the developer assumed that a target would always be sensed by multiple sensors (an assumption on physical sensing range) and hence a group would always have more than one member. The protocol had to be realigned with physical reality.

While these preliminary results are encouraging, significant challenges remain. For example, non-trivial scalability enhancements are needed. Scalability limitations imply that the designer should have some intuition into which event types and which event attributes to monitor. Since it is hard to tell which event types are relevant to a bug in advance, ideally, we would like to be able to monitor a large number of different event types and attributes, leaving it to a software tool to ignore irrelevant ones automatically, without degrading efficiency in identifying culprit event sequences. The patterns suspected of causing failures often have a false dependence on workload. For example, the frequency of occurrence of many events depends on the communication rate. If communication rates vary from one experiment to the next, discriminative sequence mining often zooms-in on differences in event traces caused by differences in communication rates and not by bugs. Some form of normalization is needed. Often one or a small number of initial (bad) events create a cascading wave, where a larger number of repercussions follow, in turn setting off an even larger number of measurable consequences (manifestations of anomalous behavior). Identifying this causal *chain* (or, in many cases, *tree*) is hard due to the difference in the frequency of occurrence of events at different levels in the tree. Unsynchronized clocks often result in finding incorrect sequences making it hard to infer distributed patterns correctly. Existence of multiple bugs often causes decreased diagnostic accuracy. Events with multiple attributes (e.g., function calls with multiple parameters) cause problems and have to be broken up into series of events with single attributes, which in turn generates false event sequences. Addressing such challenges may require interdisciplinary collaboration between data mining experts, machine learning experts, sensor networks experts, and troubleshooting experts in order to provide solutions that both consider peculiarities and requirements of sensor network troubleshooting and leverage algorithmic expertise needed for root cause diagnosis.

1.3 Data Interactions

Another important interaction space for sensor networks is the space of *data interactions*. An emerging application model for networked sensing systems is that of *social sensing* [1, 4], which broadly refers to applications that employ sensors used by individuals in their homes, cars, offices, or on their person, whose measurements may be shared for purposes of various application-related services. Social sensing systems range from medical devices that measure human biometrics and share them with medical repositories available to care-givers [29], location sensors, and accelerometers in phones and cars that can be used to compute aggregate information of community interest such as pollution or traffic patterns [9, 11, 17]. Traditional embedded and networked sensing systems research typically considers computing systems that interact with physical and engineering artifacts and assumes a single trust domain. Interaction of future embedded sensing devices with both physical and social spaces (in multiple trust domains) creates new challenges, such as loss of privacy, that can be broadly classified as *data interaction* challenges.

1.3.1 Privacy and Data Aggregation

Protection mechanisms from involuntary physical exposure are needed to enforce *physical privacy*. Innovative optimization problems can be formulated by recognizing that privacy is not a binary concept. When data are continuous and noisy, privacy is more akin to a degree of uncertainty; a concept closely related to noise and filtering in control applications. Control of voluntary information sharing must facilitate privacy-preserving exchange of time-series data. A predominant use of data in social sensing applications is for aggregation purposes such as mapping distributed phenomena or computing statistical trends. New mathematically based data perturbation and anonymization schemes are needed to hide user data but allow fusion operations on perturbed or partial data to return correct results to a high degree of approximation. These problems are difficult due to interactions between data streams.

When sharing a single data item or stream, it is easy to reason about what information is releaved and what information is withheld. When multiple streams are shared, however (possibly by different individuals), correlations between these streams may be exploited to make additional inferences that make it harder to control what exactly is being revealed. For example, sharing an acoustic energy signature from the neighborhood of a person on a campaign to reduce noise pollution may reveal something about the person's location if the noise level correlates with a train or bus schedule on a known route. This data interaction challenge makes it especially hard to reason about privacy in a social sensing system. Nevertheless, certain privacy assurances are often needed to encourage people to share the information needed for the social sensing application to function. Hence, an important challenge becomes one of understanding how to perturb (or decorrelate) data in such a way that it becomes impossible to make additional privacy-violating inferences, beyond what is explicitly allowed by the data owner.

Data aggregation operations are most common in social sensing systems where multiple data streams need to be combined to compute some community-wide information such as energy consumption trends, driving patterns, or fitness and weight loss statistics. The challenge is to perturb a user's sequence of data values such that (i) the individual data items and their trend (i.e., their changes with time) cannot be estimated without large error, whereas (ii) the distribution of the data aggregation results at any point in time is estimated with high accuracy. Intuitively, privacy in this context refers to the degree of uncertainty or error regarding a user's individual data. For instance, in a health-and-fitness application, it may be desired to find the average weight loss trend of those on a particular diet or exercise routine as well as the distribution of weight loss as a function of time on the diet. This is to be accomplished without being able to reconstruct any individual's weight and weight trend without significant error.

1.3.2 Perturbation Examples and Time-Series Data

Examples of data perturbation techniques can be found in [2, 3, 10]. The general idea is to add random noise with a known distribution to the user's data, after which a reconstruction algorithm is used to estimate the distribution of the original data. Early approaches relied on adding independent random noise. These approaches were shown to be inadequate. For example, a special technique based on random matrix theory has been proposed in [23] to recover the user data with high accuracy. Later approaches considered hiding individual data values collected from different private parties, taking into account that data from different individuals may be correlated [16]. However, they do not make assumptions on the model describing the *evolution* of data values from a given party over time, which can be used to jeopardize privacy of data streams. By developing a perturbation technique that specifically considers the data evolution model, one can prevent attacks that extract regularities in correlated data such as spectral filtering [23] and principal component analysis (PCA) [16].

In other work [11], it was shown that privacy of time-series data can be preserved if the noise used to perturb the data is itself generated from a process that approximately models the measured phenomenon. For instance, in the weight watchers example, we may have an intuitive feel for the time scales and ranges of weight evolution when humans gain or lose weight. Hence, a noise model can be constructed that exports realistic-looking parameters for both the direction and time constant of weight changes. We can think of this noise as the (possibly scaled) output of a *virtual user*. Once the noise model is available, its structure and probability distributions of all parameters are agreed upon among all parties contributing to the aggregation result. By choosing random values for these noise parameters from the specified distribution, it is possible to generate arbitrary weight curves (of virtual people) showing weight gain or loss. A real user can then add their true weight curve to that of one or several locally generated virtual users obtained from the noise model. The actual model parameters used to generate the noise are kept private. The resulting perturbed stream is shared with the pool where it can be aggregated with that of others in the community. Since the distributions of noise model parameters are statistically known, it is possible to estimate the sum, average, and distribution of added noise (of the entire community) as a function of time. Subtracting that known average noise time series from the sum of perturbed community curves will thus yield the true community trend. The distribution of community data at a given time can similarly be estimated (using deconvolution methods) since the distribution of noise (i.e., data from virtual users) is known. The estimate improves with community size.

An important question relates to the issue of trust. Given that non-expert users cannot be asked to derive good noise models for their private data, how does a non-expert client know that a given noise model is privacy-preserving? Obtaining the noise model from an external party is risky. If the party is malicious, it could send a "bad" model that is, say, a constant, or a very fast-changing function (that can be easily separated from real data using a low-pass filter), or perhaps a function with a very small range that perturbs real data by only a negligible amount. Such noise models will not perturb data in a way that protects privacy.

Consider an information aggregation service that announces a suggested noise model structure and parameter distribution to the community of users over which aggregation is performed. The model announced by the server can be trusted by a user only if that user's own data could have been generated from some parameter instantiation of that model with a non-trivial probability. This can be tested locally by a curve fitting tool on the user's side. Informally, a noise model structure and parameter distributions are accepted by a user only if (i) the curve fitting error for user's own data is not too large and (ii) the identified model parameter values for user's data (that result from curve fitting) are not too improbable (given the probability distributions of model parameters). A friendly user interface can be developed to automate the verification of the noise model on the user's behalf.

More formally, consider a particular application where an aggregation server collects data from a community to perform statistics. In previous work [11], a perturbation algorithm is described for a community of N individuals who share M data points with the aggregation each (we assume this to be the same across users for notational simplicity). Let $x^i = \left(x_1^i, x_2^i, \ldots, x_M^i\right)$, $n^i = \left(n_1^i, n_2^i, \ldots, n_M^i\right)$, and $y^i = \left(y_1^i, y_2^i, \ldots, y_M^i\right)$ represent the data stream, noise, and perturbed data shared by user i, respectively. At time instant k, let $f_k(x)$ be the empirical community distribution, $f_k^e(x)$ be the exact community distribution, $f_k(n)$ be the noise distribution, and $f_k(y)$ be the perturbed community distribution. Most user data streams can be generated according to either linear or non-linear discrete models. In general, a model can be written as a discrete function of index k, which can be time, distance, or other (depending on the application), parameters $\underline{\boldsymbol{\theta}}$ and inputs $\underline{\mathbf{u}}$, and is denoted as $g(k, \underline{\boldsymbol{\theta}}, \underline{\mathbf{u}})$. Notice that $\underline{\boldsymbol{\theta}}$ is a fixed length parameters vector characterizing the model while $\underline{\mathbf{u}}$ is a vector of length M characterizing the input to the model at each instance. Given the data $x^i = \left(x_1^i, x_2^i, \ldots, x_M^i\right)$, the model $g(k, \underline{\boldsymbol{\theta}}, \underline{\mathbf{u}})$, and the approximated distributions $f_\theta^n(\underline{\boldsymbol{\theta}})$, $f_u^n(\underline{\mathbf{u}})$, the perturbed data for user i is generated by (i) generating samples $\underline{\boldsymbol{\theta}}_{\mathbf{n}}^{\mathbf{i}}$ and $\underline{\mathbf{u}}_{\mathbf{n}}^{\mathbf{i}}$, from the distributions $f_\theta^n(\underline{\boldsymbol{\theta}})$ and $f_u^n(\underline{\mathbf{u}})$, respectively, (ii) generating noise stream $n^i = \left(n_1^i, n_2^i, \ldots, n_M^i\right)$, where $n_k^i = g\left(k, \underline{\boldsymbol{\theta}}_{\mathbf{n}}^{\mathbf{i}}, \ \underline{\mathbf{u}}_{\mathbf{n}}^{\mathbf{i}}\right)$, and (iii) generating perturbed data by adding the noise stream to the data stream $y^i = x^i + n^i$.

Now, consider reconstructing the *distribution* of community data at a given point in time. At time instance k, the perturbed data of each user are the sum of the actual data and the noise $y_k^i = x_k^i + n_k^i$. Thus, the distribution of the perturbed data $f_k(y)$ is the *convolution* of the community distribution $f_k(x)$ and the noise distribution $f_k(n)$, $f_k(y) = f_k(n) * f_k(x)$. All the distributions above can be discretized and the equation can be rewritten as: $f_k(y) = H f_k(x)$, where H is a Toeplitz cyclic matrix, which is also called the blurring kernel, constructed from the elements of the discrete distribution $f_k(n)$, $f_k(x)$ is the community distribution at time k that needs to be estimated, H is known, and $f_k(y)$ is the empirical perturbed data distribution. This problem is well known in the literature as the *deconvolution problem*. The Tikhonov–Miller restoration method [34] can be employed to compute the community distribution. It requires an a priori bound $\boldsymbol{\varepsilon}$ for the L^2 norm of the noise, together with an a priori bound M for the L^2 norm of the community distribution, $\left|\left|H f_k^e(x) - f_k(y)\right|\right|_2 \leq \boldsymbol{\varepsilon}$ and $\left|\left|(H^T H)^{-\nu} f_k^e(x)\right|\right|_2 \leq M$.

The approach preserves individual user privacy while allowing accurate reconstruction of community statistics. A multi-dimensional extension of this approach was presented in recent literature [32]. In this example, perturbation was added to the GPS trajectories of individual vehicles. The perturbed trajectories were then shared with a central server, whose responsibility was to reconstruct traffic statistics in a city. While the individuals who shared their data were allowed to "lie" about their (i) GPS location, (ii) velocity, and (iii) time of day, the reconstruction was shown to be accurate in that it reported the true average speed on city streets as a function of correct actual location and time. Figure 1.1 (reproduced from [32]) depicts the manner in which GPS trajectories are perturbed in this approach. Figure 1.2 (reproduced from [32]) compares the distribution of ground truth speed data to the reconstructed distribution obtained from perturbed shared data. It can be seen that the two distributions are rather similar. As a measure of similarity, Table 1.1 (reproduced from [32]) compares the percentages of speeding vehicles on four city streets, obtained using original and perturbed data, respectively. It can be seen that the estimates obtained from perturbed data are reasonably accurate.

Several research questions remain. For example, what is a good upper bound on the reconstruction error of the data aggregation result as a function of the noise statistics introduced to perturb the individual inputs? What are noise generation techniques that minimize the former error (to achieve accurate aggregation results) while maximizing the noise (for privacy)? How to ensure that data of individual data streams cannot be inferred from the perturbed signal? Intuitively, this is doable because traditional filtering methods such as PCA and spectral filtering work based on the assumptions that additive noise is time independent, independent of the signal, and has a small variance compared to the signal variance. With a good perturbation scheme, these assumptions are violated. What are some bounds on minimum error in reconstruction of individual data streams? What are noise generation techniques that maximize such error for privacy? Privacy challenges further include the investigation of attack models involving corrupt noise models (e.g., ones that attempt to deceive non-expert users into using perturbation techniques that do not achieve adequate privacy protection), malicious clients (e.g., ones that do not follow

Table 1.1 Percentage of speeding vehicles

Street	Real %	Reconstructed %
University Ave	15.60	17.89
Neil Street	21.43	23.67
Washington Street	0.5	0.15
Elm Street	6.95	8.6

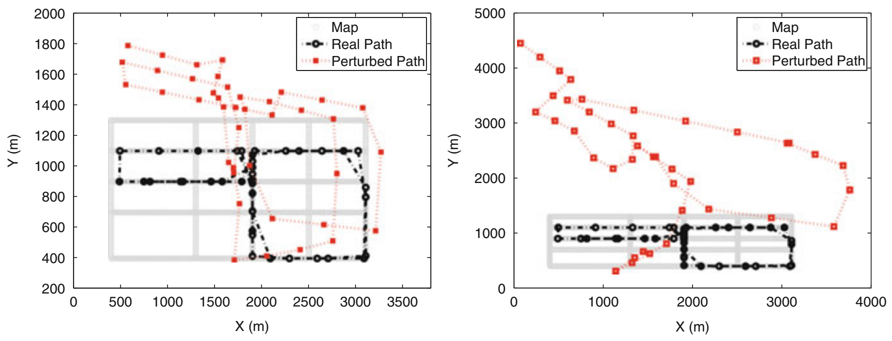


Fig. 1.1 Real and perturbed traffic trajectories for different perturbation levels

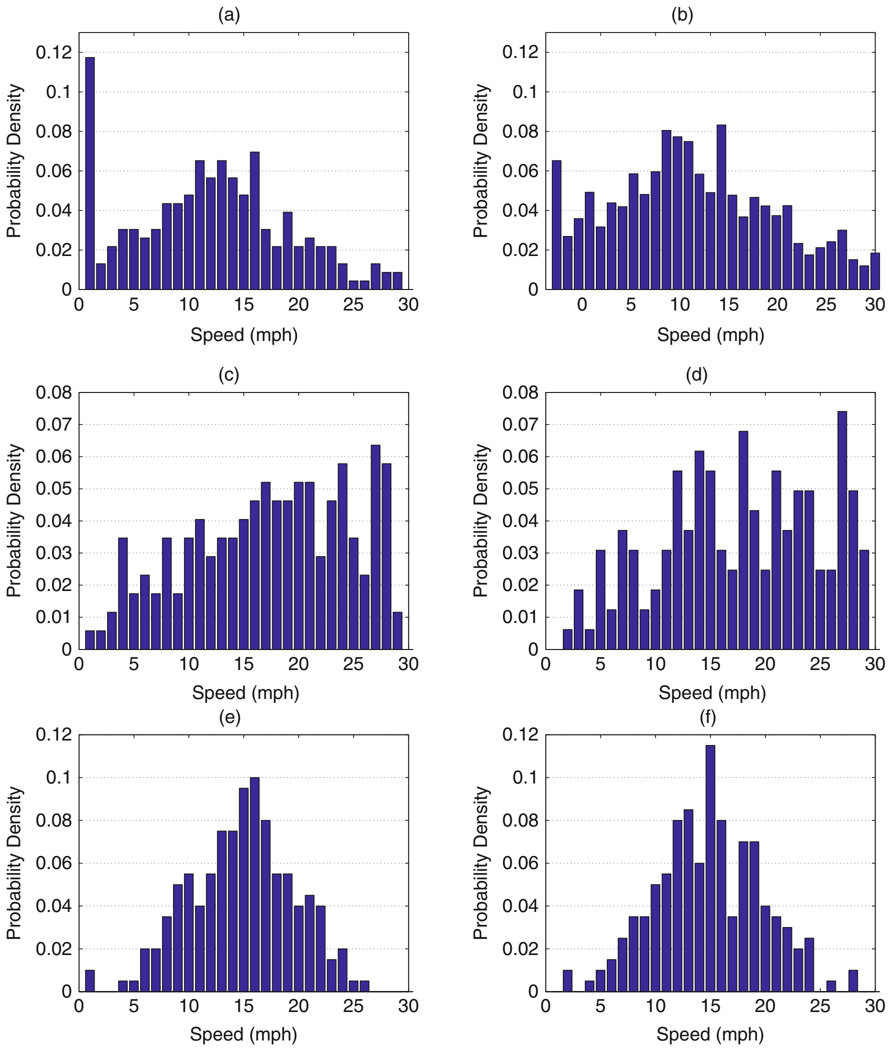


Fig. 1.2 Real and reconstructed traffic speed distributions. (**a**) Real community speed distribution; (**b**) Reconstructed speed distribution; (**c**) Real speed distribution on University Ave.; (**d**) Reconstructed speed distribution on University Ave.; (**e**) Real speed distribution on Washington Ave.; and (**f**) Reconstructed speed distribution on Washington Ave.

the correct perturbation schemes or send bogus data), and repeated server queries (e.g., to infer additional information about evolution of client data from incremental differences in query responses). For example, given that it is fundamentally impossible to tell if a user is sharing a properly perturbed version of their real weight or just some random value, what fractions of malicious users can be accommodated without significantly affecting reconstruction accuracy of community statistics? Can damage imposed by a single user be bounded using outlier detection techniques that exclude obviously malicious users? How does the accuracy of outlier detection depend on the scale of allowable perturbation? In general, how to quantify the trade-off between privacy and robustness to malicious user data? How tolerant is the perturbation scheme to collusion among users that aims to bias community statistics? Importantly, how does the *time-series* nature of data affect answers to the above questions compared to previous solutions to similar problems in other contexts (e.g., in relational databases)? The above challenges offer significant research opportunities in the area of data interactions and social sensing.

The above perturbation techniques, defense solutions, and bounds are especially challenging due to the presence of multiple correlated data streams, or data streams with related context. For example, consider a social sensing application where users share vehicular GPS data to compute traffic speed statistics in a city. In this case, in order to compute the statistics correctly as a function of time and location, each vehicle's speed must be shared together with its current GPS location and time of day. Perturbing the speed alone does not help privacy if the correct location of the user must be revealed at all times. What is needed is a perturbation and reconstruction technique that allows a user to "lie" about their speed, location, and time of day, altogether, in a manner that makes it impossible to reconstruct their true values, yet allow an aggregation service to average out the added multi-dimensional noise and accurately map the true aggregate traffic speed as a function of actual time and space. This problem is related to the more general concern of privacy-preserving classification [36, 37], except that it is applied to the challenging case of aggregates of time-series data. Understanding the relation between multi-dimensional error bounds on reconstruction accuracy and bounds on privacy, together with optimal perturbation algorithms in the sense of minimizing the former while maximizing the latter, remains an open research problem.

1.4 Temporal Interactions

The third important interaction space for computing applications that interface with the physical world is the space of *temporal interactions*. Embedded computations must generally obey not only functional integrity constraints but also timeliness constraints on results. Early applications of sensor networks focused on soft domains where it was not critical to analyze timing properties. Eventually, as the range of sensor network applications extends to include mission-criticial and safety-critical ones, the timeliness of interactions with the physical world will become important. Recent sensor network literature reflects increasing interest in analysis of real-time behavior and time constraints [6, 13, 28, 33]. Hence, theory is needed to analyze end-to-end delay in large networked sensing systems that execute a set of distributed real-time tasks.

Timing correctness requirements arise due to data volatility, interaction with mobile objects, and the need for timely reaction to environmental events. The time-sensitive nature of sensor network applications and environmental interactions motivates understanding of the real-time limitations of information transfer, such that future networks could be properly sized for the desired real-time transfer capability. Real-time information transfer is characterized by deadlines on data communication. In a hard real-time application, only those bits that are transferred prior to their deadlines contribute useful information. Missed deadlines result in adverse consequences that range from utility loss to significant physical damage. Deadlines could arise for various reasons, for example, the necessity to react to external events in a timely manner, and the need to deliver dynamically changing data prior to the expiration of their respective validity intervals.

Recently, information-theoretic bounds were derived for sensor networks that quantify the ability of the network to transfer bytes across distance [12, 30]. For time-sensitive applications, a more useful bound should also be a function of delay. Observe that network delay and throughput are interrelated. Intuitively, the network should be able to transfer more bits by their deadlines if the deadlines are more relaxed. Finding a function that bounds achievable total capacity subject to delay constraints is a new objective that has not been addressed in sensor network literature. We call it *real-time capacity* of distributed sensing systems.

1.4.1 Temporal Analysis of Distributed Systems

As a step toward a fundamental understanding of real-time capacity, significant advances were independently made in the embedded systems and networking communities to quantify the timing properties of distributed computation or communication. Existing techniques for analyzing delay/throughput trade-offs in distributed systems can be broadly called *decomposition based*. Decomposition-based techniques break the system into multiple subsystems, analyze each subsystem independently, then combine the results. Network calculus [7, 8] (developed in the networking community) and holistic analysis [31, 35] (developed in the embedded systems community) fall into this category.

A deficiency with decomposition-based approaches is that by viewing the aggregate problem as a set of smaller subproblems, often interactions between the subproblems are ignored or simplified. For example, network calculus does not accurately account for the effects of pipelining between stages when multiple flows share the same set of successive hops. It merely computes a delay bound on each hop based on its service curve and its arrival curve (computed from the service curve on the previous hop). Pipelining, however, makes it impossible for the same flow to suffer the delay bound on several successive hops in a row. Intuitively, this is because if a packet waits for many other packets to get processed on one hop, by the time its turn comes, most of the other packets have already moved sufficiently downstream that they may not interfere with it again. Hence, this packet's interference at the downstream hop is lower than what the bound predicts. This is an example of a temporal interaction between stages in a distributed system that leads to a *sub-additive property* of individual (i.e., per hop) worst-case delay bounds. Generally speaking, the worst-case delay of a task on two successive stages of processing is *deterministically less* than the sum of its worst-case delays on the individual stages.

An exception is the top priority task that suffers no interference on its path. There is a need to quantify these subadditive delay properties of distributed computation. At present, very little work exists toward a general theory for delay composition and the relation of delay composition results to priorities of tasks and the amount of present interference.

Consider, by contrast, the fundamental laws of circuit theory used to analyze linear electric circuits. In that theory, a small number of fundamental rules (e.g., Kirchhoff laws) allow a designer to analyze complex circuits of arbitrary interconnection topology, reducing them to their effective transfer functions and deducing their exact end-to-end characteristics, such as total impedance, current draw, and voltage drop. The same compositionality is observed in feedback control theory, where component models, represented by block diagrams, can be collapsed into an equivalent single block that accurately expresses the overall system model and enables controller design. A similar theory is needed for networked sensing systems that develop rules for composition of temporal behavior of real-time system components. We call this category of techniques for analyzing distributed systems *reduction-based* (as opposed to decomposition-based) techniques. It is key for reductions to capture the essential properties of components involved and the properties of their interactions. This ensures that reductions do not lead to inaccuracies caused by ignored or oversimplified dependencies.

1.4.2 Reduction-Based Analysis and Delay Composition Algebra

A recent reduction-based approach to composition of timing properties of distributed sensing systems is delay composition algebra [21]. Given a graph of system resources, where nodes represent processing resources and arcs represent the direction of job flow, algebraic operators systematically "merge" resource nodes, composing their workloads per rules of the algebra, until only one node remains. The workload of that node represents a single resource job set called the *uniprocessor job set*. Uniprocessor schedulability analysis can then be used to determine the schedulability of the set.

Workload of any one node (that may represent a single resource or the result of reducing an entire subsystem) is described generically by a two-dimensional matrix stating the worst-case delay that each job, J_i, imposes on each other job, J_k, in the subsystem the node represents. Let us call it the *load matrix* of the subsystem in question. Observe that on a node that represents a single resource j, any job J_i that is of higher priority than job J_k can delay the latter by at most J_i's worst-case computation time, $C_{i,j}$, on that resource. This allows one to trivially produce the load matrix for a single resource given job computation times, $C_{i,j}$, on that resource. Element (i,k) of the load matrix for resource j, denoted $q^j_{i,k}$ (or just $q_{i,k}$ for notational simplicity where no ambiguity arises), is simply equal to $C_{i,j}$ as long as J_i is of (equal or) higher priority than J_k. It is zero otherwise.

The main question becomes, in a distributed system, how to compute the worst-case delay that a job imposes on another when the two meet on more than one resource? The answer decides how delay components of two load matrices are combined when the resource nodes corresponding to these matrices are merged using appropriate algebraic operators. Intuitions derived from single resource systems suggest that delays are combined *additively*. This is not true in distributed systems. In particular, it was shown in [20] that worst-case delays in pipelines are *subadditive* because of gains due to parallelism caused by pipelining. More specifically, the worst-case delay imposed by a higher priority job, J_i, on a lower priority job, J_k, when both traverse the same set of stages varies with the *maximum* of J_i's per-stage computation times, not their *sum* (plus another component we shall mention shortly).

The delay composition algebra leverages the aforementioned result. Neighboring nodes in the resource DAG present an instance of pipelining, in that jobs that complete execution at one node move on to execute at the next. Hence, when these neighboring nodes are combined, the delay components, $q_{i,k}$, in their load matrices are composed by a maximization operation. In delay composition algebra, this is done by the PIPE operator. It reduces two neighboring nodes to one and combines the corresponding elements, $q_{i,k}$, of their respective load matrices by taking the maximum of each pair. For this reason, we call $q_{i,k}$ the *max term*.

It could be, however, that two jobs travel together in a pipelined fashion for a few stages (which we call a pipeline segment), then split and later merge again for several more stages (i.e., another pipeline segment). Consider a higher priority job J_i and a lower priority job, J_k. In this case, the max terms of each of the pipeline segments (computed by the maximization operator) must be added up to compute the total delay that J_i imposes on J_k. It is convenient to use a running counter or "accumulator" for such addition. Whenever the jobs are pipelined together, delays are composed by maximization (kept in the max term) as discussed above. Every time J_i splits away from J_k, signaling the termination of one pipeline segment, the max term (i.e., the delay imposed by J_i on J_k in that segment) is added to the accumulator. Let the accumulator be denoted by $r_{i,k}$. Hence, $r_{i,k}$ represents the total delay imposed by J_i on a lower priority job J_k over all past pipeline segments they shared. Observe that jobs can split apart only at those nodes in the DAG that have more than one outgoing arc. Hence, in the algebra, a SPLIT operator is used when a node in the DAG has more than one outgoing arc. SPLIT updates the respective accumulator variables, $r_{i,k}$, of all those jobs J_k, where J_k and a higher priority job J_i part on different arcs. The update simply adds $q_{i,k}$ to $r_{i,k}$ and resets $q_{i,k}$ to zero.

In summary, in a distributed system, it is useful to represent the delay that one job J_i imposes on another J_k as the sum of two components $q_{i,k}$ and $r_{i,k}$. The $q_{i,k}$ term is updated upon PIPEs using the maximization operator (the max term). The $r_{i,k}$ is the accumulator term. The $q_{i,k}$ is added to the $r_{i,k}$ (and reset) upon SPLITs, when J_i splits from the path of J_k. PIPE and SPLIT are thus the main operators of the algebra. In the final resulting matrix, the $q_{i,k}$ and $r_{i,k}$ components are added to yield the total delay that each job imposes on another in the entire system.

The final matrix is indistinguishable from one that represents a uniprocessor task set. In particular, each *column k* in the final matrix denotes a uniprocessor job set of jobs that delay J_k. In this column, each non-zero element determines the computation time of one such job J_i. Since the transformation is agnostic to periodicity, in the case of periodic tasks, J_i and J_k simply represent the parameters of the corresponding periodic task invocations. Hence, for any task, T_k, in the original distributed system, the final matrix yields a uniprocessor task set (in column k), from which the schedulability of task T_k can be analyzed using uniprocessor schedulability analysis.

Finally, the above discussion omitted the fact that the results in [20] also specified a component of pipeline delay that grows with the number of stages traversed by a job and is independent of the number of higher priority jobs, called the *stage-additive* component, s_k. Hence, the load matrix, in fact, has an extra row to represent this component. As the name suggests, when two nodes are merged, this component is combined by addition. A detailed account of delay composition algebra, including a complete exact specification of its operators and examples of its use, can be found in [20–22]. One can easily envision examples from the sensor networks domain, where aggregation trees, for instance, lead to traffic patterns where transmission of individual flows (represented as a pipeline of forwarding stages) forms a DAG or convergecast graph, whose end-to-end delay may need to be bounded. Other examples include query processing applications, where a single query may be divided across multiple nodes to be evaluated against different subsets of data, then the results combined. Given multiple queries of different query processing graphs, their end-to-end timing behavior can be analyzed using the above approach.

Delay composition algebra is a step toward understanding temporal interactions and composition of timing properties in distributed sensing systems. In turn, this understanding can lead to a quantification of new notions of real-time capacity. Several questions must be answered for a useful real-time capacity theory to emerge:

- *Load metrics:* Real-time capacity must be expressed in appropriate load metrics. For example, classical schedulability bounds are expressed in terms of utilization. For distributed systems, one must determine which of the family of viable load metrics is the "best" metric to use to quantify the ability of the system to meet timing constraints.
- *Sufficient capacity regions:* Real-time capacity quantifies system load that can be supported within time constraints, which known as the *schedulability* problem. Schedulability, however, is an NP-hard problem and gives rise to very complex (porcupine) scheduable state spaces. To derive practical analytic capacity expressions, sufficient schedulability conditions must be found, meaning those defined by simple surfaces that encompass most (but not necessarily all) states in which timing constraints are met. We call them capacity regions. There is an inherent trade-off between the simplicity of capacity regions and their degree of approximation. Good compromises must be sought that maintain simplicity without introducing excessive pessimism.
- *Composition rules:* Rules must be defined for composing capacity regions of large systems from those of their subsystems. In general, starting with capacity regions of elementary components, one should be able to compose capacity regions of arbitrarily large systems. Most importantly, capacity expressions should not become more pessimistic with composition. For very large systems,

where the number of components can be viewed as infinite, continuous forms of composition rules are needed. Composition becomes an integration operation over functions of component densities as opposed to an operation that is carried out on individual components. Delay composition algebra is a first step toward defining such composition rules.

- *Optimization algorithms:* Capacity regions define sets of system states that meet sufficient time constraints. It may be desired to optimize various metrics within those constraints. For example, one might want to derive points of maximum sensor network throughput or minimum total energy consumption within capacity region boundaries.

A new real-time capacity theory should make it possible to understand the timing behavior of large real-time sensing networks with in-network computation at intermediate hops. It should also become possible to quantify end-to-end behavior of complex distributed sensing applications such as distributed power grid control and telepresence. The theory should help understand how prioritization affects real-time capacity. It will be possible, for example, to do a cost/benefit analysis of prioritizing different sensor data queues in a complex distributed sensing application since the theory will quantify the effect of prioritization on the load/timeliness trade-off. The needed theory is different from previous foundations for analysis of network delay that consider networks as graphs of links that carry packets in that the role of computation on network nodes must be considered together with communication. The real-time capacity is a general notion that does not make limiting assumptions on the types of processing resources involved. Hence, both network transmissions and CPU processing should be analyzable within the same framework. This framework is needed to understand the end-to-end timing behavior of large systems involving tightly intertwined computation, communication, and sensing.

1.5 Interactions of System Dynamics

The final interaction space for distributed sensing systems addressed in this chapter is the space of system dynamics. Control engineers are trained to analyze dynamics of physical and embedded systems and verify their adherence to desired specifications. Unfortunately, dynamics (in a control-theoretic sense) are not a term that computer scientists normally come in contact with in their education. As a result, dynamics of feedback loops that pervade the design of computing software are often poorly accounted for and poorly understood. While it is easy to use simple heuristics to ensure the stability of feedback loops within smaller subsystems, unexpected consequences may arise when such subsystems are combined.

1.5.1 Sources of Dynamics in Software

Informally, software dynamics occur when systems involve "delayed" or "cumulative" response that may be approximated by differential or difference equation models. An example of software features that generate dynamics is an action taken by one system component that depends on results of another action taken by another component in the past. Another example is when some action or system parameter depends on a previously accumulated value of another parameter. For example, the number of packets waiting in a network queue depends on the accumulation (i.e., integral) of differences between past enqueue and dequeue rates. This dependency on previous values can be captured by a difference equation creating software dynamics.

If all causal responses within a software system were instantaneous, the system becomes strictly reactive in that it instantly reaches a state that is a function of only currently applied stimuli. However, in most systems, effects depend not only on current state but also on previous states. This is especially true in sensor networks, where queues and other communication delays create significant dependencies on past states. Hence, analysis of dynamics is needed. This analysis is especially critical in computing systems when feedback is used. Dynamics imply that software decisions are made based on *past* information (e.g., due to delays in acquiring or communicating the information) or that effects of actions are not immediately observed (for example, a reduction in source sending rates in a congested network will take some time before it diffuses network delays). If software feedback loops do not properly address dynamics, they may "under-" or "over-react." For example, sending rates might not be decreased enough to eliminate congestion, or conversely might be cut too much, thereby unnecessarily degrading performance. *Stability* is the property of a feedback loop that allows it to converge over time to desired performance. Control theory allows designers to analyze stability, convergence rate, overshoot, and other dynamic response properties of computing systems. In particular, control theory explains that while individual components may be stable, their composition may not be necessarily so. Hence, using ad hoc techniques in designing feedback in software systems may result in components that work well in isolation, but have poor performance when combined.

The above discussion suggests that composing, analyzing, controlling, and optimizing performance of large-scale networked sensing systems is an important problem, complicated by increased system size, a growing number of tunable parameters (and hence feedback loops that tune them), subtle interactions among distributed components, and limited observability of internal software state at run-time.

The problem is of growing importance. The increasing cost of managing large systems suggests that sensor networks and the information processing systems they serve will operate with progressively less human oversight. The trend toward increasingly autonomous, larger, and more interconnected systems exacerbates the problem in two important ways:

- First, autonomy implies increasing need for *adaptive* or *self-tuning* behavior. Many aspects of system functionality will be automated, creating a large number of feedback loops. For example, MAC-layer algorithms may automatically determine the best line transmission rates such that reliability is maximized. Routing may automatically determine the least-cost routes as load on different network components changes. Congestion control may automatically determine the best application sending rate to prevent bottlenecks and overload. Application knobs such as fidelity of information processing may be manipulated depending on factors such as currently available energy or user demand.

- Second, increased system scale implies interactions among a larger number of components, which makes component composition a growing problem. As developer teams that build software systems grow, each developer becomes responsible for a progressively smaller fraction of the system, essentially leading to myopic design. Unintended interactions among different feedback loops in such a design can lead to unexpected effects on aggregate performance. Individually designed adaptive or automated modules with efficient performance management policies (when considered in isolation) might contribute to significant performance degradation when put together. Research and management tools are needed to address these performance composability problems, especially when designers and operators do not have the analytic background to analyze overall dynamics and stability of their systems.

1.5.2 Examples of Dynamic Interactions

To give an example of adverse interactions and illustrate the importance of addressing composability of dynamic behavior in the context of distributed sensing systems, consider a scenario drawn from the domain of communication protocols. Let shortest path routing be one policy that constantly discovers shorter routes between sources and destinations. Let the MAC-layer rate adaptation policy, on the other hand, tune the radio transmission rate to match channel quality (a lower rate is used on lower quality channels). While each policy is individually well motivated, composing the two policies leads to an adverse interaction. Shortest path routing may prefer longer hops (so there are fewer of them on the path). Longer hops tend to have lower quality, which causes the radio to lower its transmission rate. At the lower rate, new more distant neighbors may be discovered leading to shorter routes. Switching to those routes reduces channel quality again, leading to further rate reductions. This adverse feedback cycle ultimately diminishes throughput. Such composition problems are expected to increase in software systems as these systems become more complex (i.e., made of more components) and feature more capabilities for adaptation.

Interestingly, adverse interactions may result even when the different adaptive policies have the same objective. These unintended interactions stem from subtle incompatibilities between their performance management mechanisms. Consider a distributed data processing back-end that performs multistage data fusion for a large sensing system. Two mechanisms are installed to save energy during off-peak load conditions. The first mechanism is to power off those machines that are underutilized and distribute their load across other machines in their tier. When all machines exhibit high utilization, extra machines are powered on. We call it the *On/Off policy*. The second mechanism is to employ dynamic voltage scaling (DVS) on individual processors such that the speed and voltage of a machine are reduced when the

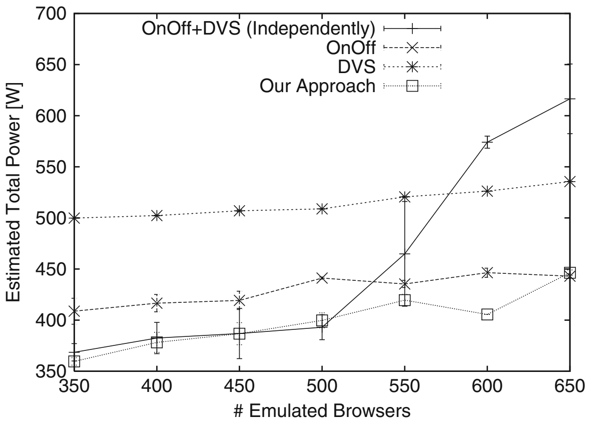


Fig. 1.3 Two adaptation policies in a multi-tier server farm and their combination

machine is underutilized and increased when it is overloaded. We call it the *DVS policy*. The two policies work well in isolation. Previous literature reports [14] that the two policies combined may actually result in a *higher* energy consumption than when one policy is used in isolation. This effect is shown in Fig. 1.3.

The explanation lies in unmodeled dynamics. If the DVS policy is aggressive enough, whenever the utilization of a machine decreases, the policy reduces clock frequency (and voltage) thus slowing down the machine and restoring a high utilization value. From the perspective of the (DVS-oblivious) On/Off policy, the farm becomes "fully utilized," as the measured utilization of all machines is high. This drives the On/Off policy to needlessly turn machines on in an attempt to relieve the full utilization condition. DVS will slow down the clock further, causing more machines to be turned on, and so on. Figure 1.3 also shows that proper joint control of both knobs (labeled "our approach" in the figure) does improve performance over tuning either knob in isolation.

To uncover unintended loops, a formal analysis of the system should use stability notions from control theory. A simplified analysis technique, based on the notion of adaptation graphs, was presented in previous computing literature [14]. Nodes in an adaptation graph represent the key variables in the system such as delay, throughput, utilization, length of different queues, and settings of different policy knobs. Arcs represent the direction of causality. For example, consider a back-end data server that serves queries over a network. When the utilization, U, of the outgoing link increases, the delay, D, of served requests increases as well (because they wait longer to be sent over the congested link). Hence, an arc exists from utilization to delay, $U \rightarrow D$, indicating that changes in the former affect the latter. The arcs in the adaptation graph are annotated by either a "+" or a "−" sign depending on whether the changes are in the same direction or not. In the example at hand, since an increase in utilization causes a *same-direction* change in delay (i.e., also an increase), the arc is annotated with a "+" sign: $U \rightarrow^{+} D$. Some of the arcs represent fundamental natural phenomena (for example, an increase in delay is a natural consequence of an increase in utilization). Others represent *programmed* behavior or policies. For example, an admission controller may be programmed to decrease the fraction of admitted requests, R, in response to an increase in delay, D. Hence, an arc exists in the adaptation graph from delay to admitted requests, $D \rightarrow^{-} R$. The arc is annotated with a "–" sign because an increase in delay results in a change in the opposite direction (i.e., a decrease) in admitted requests. This arc does not represent a natural phenomenon but rather the way the admission control policy is programmed. These arcs are called *policy arcs* and annotated with the name of the module implementing the corresponding policy. Hence, we have $D \rightarrow^{-}_{\mathrm{AC}} R$, where AC stands for the admission control module. Figure 1.4a depicts the adaptation graph of the data server under consideration. The graph is composed of three arcs. The arc $D \rightarrow^{-}_{\mathrm{AC}} R$ reflects that the admission controller reduces the number of admitted requests when delay increases and vice versa. The arc $R \rightarrow^{+} U$ reflects the natural phenomenon that any changes in the number of admitted requests result in same-direction changes in outgoing link utilization. Finally, the arc $U \rightarrow^{+} D$ expresses the fact that changes in link utilization cause changes in delay (in the same direction). The three arcs form a cycle (a feedback loop). An interesting property of the loop is that the product of the signs of the arcs is negative. This indicates a negative feedback loop, which is expected for stability.

As another example, consider a network power management middleware that measures links utilization, U. If the link utilization is low, the server workload must be low. The middleware thus engages dynamic voltage scaling (DVS) on the server to lower processor voltage, V, and frequency, F, hence reducing power consumption, P, due to the off-peak load condition. This adaptation action can be expressed as $U \rightarrow^{+}_{\mathrm{PM}} V$ and $U \rightarrow^{+}_{\mathrm{PM}} F$, where PM stands for power management middleware (i.e., a decrease in link utilization causes the policy to decrease both voltage and frequency which explains the signs on the arcs). In turn, we have $V \rightarrow^{+} P$ and $F \rightarrow^{+} P$, which says how power consumption changes with voltage and frequency. Finally, we have $F \rightarrow^{-} D$, since lowering frequency (i.e., slowing down a processor) increases delay and vice versa. Figure 1.4b depicts the adaptation graph for the network power management middleware.

As might be inferred from above, each component or subsystem of a larger system has its own adaptation graph that describes what performance variables this component is affecting and what causality chains (or loops) exist within. When a system is composed, the adaptation graphs of individual components are coalesced. Figure 1.4c shows the combined adaptation graph that results when a server described in Fig. 1.4a operates on top of the middleware described in Fig. 1.4b. To check for incompatibilities (adverse interactions), the graph is searched for loops using any common graph traversal algorithm. Loops that traverse component boundaries are emergent behavior loops that have not been created by design. In particular, if the product of signs on one such loop is positive, the cycle indicates an unsafe feedback loop. In other words, a stimulus reinforces itself causing more change in the same direction. In control-theoretic terms, such a loop is *unstable*.

For example, in Fig. 1.4c, the cycle $U \rightarrow^{+}_{\mathrm{PM}} F$, $F \rightarrow^{-} D$, $D \rightarrow^{-}_{\mathrm{AC}} R$, $R \rightarrow^{+} U$ crosses module boundaries and has positive sign product, indicating that it is unstable. The cycle is an instance of an adverse interaction explained as

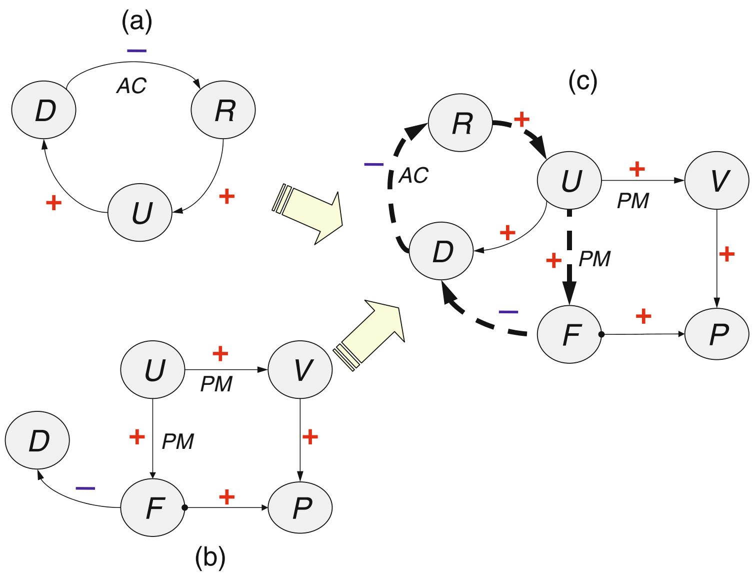


Fig. 1.4 Examples of adaptation graphs and their combination. (**a**) Adaptation graph of an admission controller of a performance-aware server; (**b**) Adaptation graph of a network power management middleware; and (**c**) Combined adaptation graph of the two

follows. Starting with the node labeled, U, when the network utilization decreases in the server, the power management middleware causes the server to slow down. This, in turn, increases the delay experienced by served requests causing the admission controller to accept fewer requests. The reduced accepted number of requests will further decrease the load on the network link, causing the power management middleware to slow down processors even more. This, in turn, may cause a more significant reduction in admitted requests and a further reduction in network load. This cycle could ultimately bring the server to a crawl, indeed an adverse consequence of unintended interaction.

Analytic foundations and tools are needed for the design, composition, and optimization of performance of large-scale distributed, adaptive, sensing systems. Much of our future infrastructure, such as power grids, homeland defense systems, and disaster recovery systems will likely be able to make use of insights and contributions of such a theory. It should be noted that despite the promise of control-theoretic techniques in analysis of system dynamics, they fall short of analysis of networked sensing systems. This is because computing systems offer new nonlinearities and different functionalities not adequately modeled by linear difference equations. Hence, extensions are needed to non-linear control to address the specific nonlinear and functional behaviors common to networked sensing systems in order to reason about their closed loop behavior. Such techniques must further be scaled to predict emergent behavior of large highly interconnected, interacting systems, as opposed to analyzing performance of isolated feedback loops.

1.6 Summary

This chapter described some practical considerations in the design of large networked sensing systems that arise in different interaction spaces between system components. Functional, data, temporal, and dynamic interaction spaces were explored. It was shown that new challenges arise in handling problems that occur by virtue of scale. Problems and interactions addressed in this chapter do not typically manifest themselves in smaller deployments. Tools and techniques are needed for sensor network designers to address the above composition and scaling challenges.

References

1. T. F. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. J. Guibas, A. Kansal, S. Madden, and J. Reich. Mobiscopes for human spaces. *IEEE Pervasive Computing*, 6(2):20–29, 2007.
2. D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the 20th ACM SIGMOD Symposium on Principles of Database Systems*, pages 247–255, Santa Barbara, CA, 2001.
3. R. Agrawal and R. Srikant. Privacy preserving data mining. In *Proceedings of ACM Conference on Management of Data*, pages 439–450, Dallas, TX, May 2000.
4. J. Burke et al. Participatory sensing. Workshop on World-Sensor-Web, co-located with ACM SenSys, 2006.
5. Q. Cao, T. Abdelzaher, J. Stankovic, and T. He. LiteOS, a UNIX-like operating system and programming platform for wireless sensor networks. In *IPSN/SPOTS*, St. Louis, MO, April 2008.
6. K. Chintalapudi and L. Venkatraman. On the design of mac protocols for low-latency hard real-time discrete control applications over 802.15.4 hardware. In *IPSN/SPOTS*, St. Louis, Missouri, 2008.
7. R. Cruz. A calculus for network delay, part i: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.
8. R. Cruz. A calculus for network delay, part ii: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, January 1991.
9. S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell. The bikenet mobile sensing system for cyclist experience mapping. In *SenSys '07: Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, pages 87–101, ACM, New York, NY, USA, 2007.
10. A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the SIGMOD/PODS Conference*, pages 211–222, San Diego, CA, 2003.
11. R. K. Ganti, N. Pham, Y.-E. Tsai, and T. F. Abdelzaher. Poolview: Stream privacy for grassroots participatory sensing. In *Proceedings of SenSys '08*, pages 281–294, Raleigh, NC, 2008.
12. P. Gupta and P. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.
13. T. He, B. M. Blum, Q. Cao, J. A. Stankovic, S. H. Son, and T. F. Abdelzaher. Robust and timely communication over highly dynamic sensor networks. *Journal of Real-time Systems*, 37(3):261–289, December 2007.
14. J. Heo, D. Henriksson, X. Liu, and T. Abdelzaher. Integrating adaptive components: An emerging challenge in performance-adaptive systems and a server farm case-study. In *Real-Time Systems Symposium*, Tuscon, AZ, December 2007.
15. J.-H. Huang, S. Amjad, and S. Mishra. Cenwits: a sensor-based loosely coupled search and rescue system using witnesses. In *Proceedings of SenSys*, pages 180–191, San Diego, CA, 2005.

16. Z. Huang, W. Du, and B. Chen. Deriving private information from randomized data. In *Proceedings of the 2005 ACM SIGMOD Conference*, pages 37–48, Baltimore, MD, June 2005.
17. B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. K. Miu, E. Shih, H. Balakrishnan, and S. Madden. CarTel: A Distributed Mobile Sensor Computing System. In *4th ACM SenSys*, Boulder, CO, November 2006.
18. B. Hull et al. Cartel: a distributed mobile sensor computing system. In *Proceedings of SenSys*, pages 125–138, 2006.
19. C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1):2–16, 2003.
20. P. Jayachandran and T. Abdelzaher. A delay composition theorem for real-time pipelines. In *ECRTS*, pages 29–38, Pisa, Italy, July 2007.
21. P. Jayachandran and T. Abdelzaher. Delay composition algebra: A reduction-based schedulability algebra for distributed real-time systems. In *RTSS*, pages 259–269, Barcelona, Spain, December 2008.
22. P. Jayachandran and T. Abdelzaher. Transforming acyclic distributed systems into equivalent uniprocessors under preemptive and non-preemptive scheduling. In *ECRTS*, Prague, Czech Republic, July 2008.
23. H. Kargutpa, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the IEEE International Conference on Data Mining*, pages 99–106, Melbourne, Florida, 2003.
24. M. Khan, T. Abdelzaher, and K. Gupta. Towards diagnostic simulation in sensor networks. In *DCoSS*, Santorini, Greece, June 2008.
25. M. Khan, T. Abdelzaher, and L. Luo. SNTS: Sensor network troubleshooting suite. In *DCoSS*, Santa Fe, NM, June 2007.
26. M. Khan, H. K. Le, H. Ahmadi, T. Abdelzaher, and J. Han. Dustminer: Troubleshooting interactive complexity bugs in sensor networks. In *ACM Sensys*, Raleigh, NC, November 2008.
27. H. K. Le, D. Henriksson, and T. F. Abdelzaher. A practical multi-channel media access control protocol for wireless sensor networks. In *IPSN*, pages 70–81, St. Louis, Missouri, 2008.
28. H. Li, P. Shenoy, and K. Ramamritham. Scheduling messages with deadlines in multi-hop real-time sensor networks. In *11th IEEE Real Time and Embedded Technology and Applications Symposium*, pages 415–425, San Francisco, CA, March 2005.
29. Microsoft Health Vault. http://www.healthvault.com/. Accessed on November 30, 2010.
30. T. Moscibroda. The worst-case capacity of wireless sensor networks. In *IPSN*, pages 1–10, Cambridge, MA, 2007.
31. J. Palencia and M. G. Harbour. Schedulability analysis for tasks with static and dynamic offsets. In *IEEE Real-Time Systems Symposium*, pages 26–37, Madrid, Spain, December 1998.
32. N. Pham, R. Ganti, M. Y. Uddin, S. Nath, and T. Abdelzaher. Privacy-preserving reconstruction of multidimensional data maps in vehicular participatory sensing. In *European Conference on Wireless Sensor Networks (EWSN)*, Coimbra, Portugal, February 2010.
33. J. Song, S. Han, A. Mok, D. Chen, M. Lucas, and M. Nixon. Wirelesshart: Applying wireless technology in real-time industrial process control. In *11th IEEE Real Time and Embedded Technology and Applications Symposium*, pages 377–386, April 2008.
34. A. N. Tikhonov and V. Y. Arsenin. *Solution of Ill Posed Problems*. V. H. Winstons and Sons, 1977.
35. K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Elsevier Microprocessing and Microprogramming*, 40(2–3):117–134, 1994.
36. Z. Yang, S. Zhong, and R. N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *Proceedings of SIAM International Conference on Data Mining*, pages 92–102, Newport Beach, CA, 2005.
37. N. Zhang, S. Wang, and W. Zhao. A new scheme on privacy-preserving data classification. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 374–383, ACM, New York, NY, USA, 2005.

Part II
Models, Topology, Connectivity

Chapter 2
Scheduling and Power Assignments in the Physical Model

Alexander Fanghänel and Berthold Vöcking

Abstract In the interference scheduling problem, one is given a set of n communication requests each of which corresponds to a sender and a receiver in a multipoint radio network. Each request must be assigned a power level and a color such that signals in each color class can be transmitted simultaneously. The feasibility of simultaneous communication within a color class is defined in terms of the signal to interference plus noise ratio (SINR) that compares the strength of a signal at a receiver to the sum of the strengths of other signals. This is commonly referred to as the "physical model" and is the established way of modeling interference in the engineering community. The objective is to minimize the schedule length corresponding to the number of colors needed to schedule all requests. We study *oblivious power assignments* in which the power value of a request only depends on the path loss between the sender and the receiver, e.g., in a linear fashion. At first, we present a *measure of interference* giving lower bounds for the schedule length with respect to linear and other power assignments. Based on this measure, we devise distributed scheduling algorithms for the linear power assignment achieving the minimal schedule length up to small factors. In addition, we study a power assignment in which the signal strength is set to the square root of the path loss. We show that this power assignment leads to improved approximation guarantees in two kinds of problem instances defined by directed and bidirectional communication request. Finally, we study the limitations of oblivious power assignments by proving lower bounds for this class of algorithms.

2.1 Introduction

Simultaneously transmitted radio signals interfere with each other. Early theoretical approaches (see, e.g., [11, 13, 17]) about scheduling signals or packets in radio networks resort to *graph-based vicinity models* (also known as *protocol model*) of

A. Fanghänel (✉)
RWTH Aachen University, Aachen, Germany
e-mail: fanghaenel@cs.rwth-aachen.de

the following flavor. Two nodes in the radio network are connected by an edge in a communication graph if and only if they are in mutual transmission range. Interference is modeled through independence constraints: If a node u transmits a signal to an adjacent node v, then no other node in the vicinity of v, e.g., in the one- or two-hop neighborhood, can transmit. The problem with this modeling approach is that it ignores that neither radio signals nor interference ends abruptly at a boundary.

Recent theoretical studies [1–4, 6, 7, 9, 14, 15] use a more realistic model, the so-called *physical model*, which is well accepted in the engineering community. It is assumed that the strength of a signal diminishes with the distance from its source. More specifically, let $d(u, v)$ denote the distance between the nodes u and v. We assume the *path loss radio propagation* model, where a signal sent by node u with power p is received at node v with $p/d(u, v)^\alpha$, where $\alpha \geq 1$ is parameter of the model, the so-called *path loss exponent*.[1] A signal sent with power p by node u is received by node v at a strength of $p/d(u, v)^\alpha$. Node v can successfully decode this signal if its strength is relatively large in comparison to the strength of other signals received at the same time. This constraint is described in terms of the *signal to interference plus noise ratio (SINR)* being defined as the ratio between the strength of the signal that shall be received and the sum of the strengths of signals simultaneously sent by other nodes (plus ambient noise). For successfully receiving a signal, it is required that the SINR is at least β with $\beta > 1$ being the second parameter of the model, the so-called *gain*.

Let us illustrate the physical model with a simple but intriguing example showing the importance of choosing the right power assignment. Suppose there are two pairs of nodes (u_1, v_1) and (u_2, v_2). Two signals shall be sent simultaneously, one from u_1 to v_1 and the other from u_2 to v_2. Suppose the nodes are placed in a nested fashion on a line, that is, the points are located on the line in the order u_1, u_2, v_2, v_1 such that the distance between u_1 and u_2 is two, the distance between u_2 and v_2 is one, and the distance between v_2 and v_1 is two (cf. Fig. 2.1). For simplicity fix $\alpha = 2$ and $\beta = 1$ and neglect the noise.

- At first, let us assume that both u_1 and u_2 send their signal with the same power 1. Then the strength of u_1's signal at node v_1 is $1/25$ while the strength of u_2's signal at the same node is $1/9$. Hence, v_1 cannot decode the signal sent by node u_1 as it is drowned by u_2's signal. That is, the outer pair is blocked by the inner pair when using uniform powers.
- At second, let us assume that signals are sent in a way that the path loss is compensated, that is, both nodes use a strength that is linear in the path loss. In particular, u_1 sends at power 25 and u_2 sends at power 1. Now consider the strengths of the signals received at v_2: The strength of u_2's signal is only 1 while the strength of u_1's signal is $25/9$. Thus, the inner pair is blocked by the outer pair when using powers that are chosen linear in the path loss.
- Finally, let us make an attempt setting the powers equal to the square root of the path loss, that is, u_1 uses power 5 and u_2 uses power 1. Now easy calculus

[1] Depending on the environment, it is usually assumed that α has a value between 2 and 5.

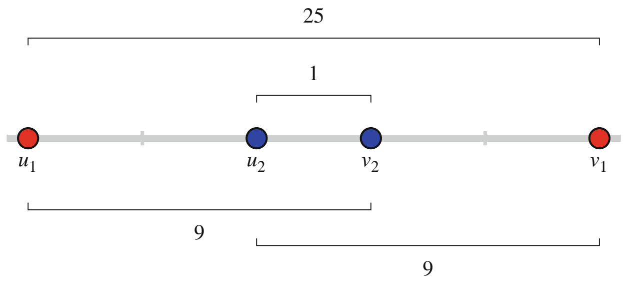


Fig. 2.1 Placement of the nodes and the path loss for $\alpha = 2$. Linear and uniform power assignment both need different schedule steps for each of the requests, the square root power assignment can schedule both requests at once

shows that, at v_1, the strength of u_1's signal is larger than the strength of u_2s and, at v_2, the strength of u_2's signal is larger than the strength of u_1s. Hence, simultaneous communication between the nested pairs is possible when choosing the right power assignment.

In this chapter, we investigate interference scheduling problems like the one in the example above. In general, one is given a set of n communication requests, each consisting of a pair of points in a metric space. Each pair shall be assigned a power level and a color such that the pairs in each color class can communicate simultaneously at the specified power. The feasibility of simultaneous communication within a color class is described by SINR constraints. The objective is to minimize the number of colors, which corresponds to minimizing the time needed to schedule all communication requests. As this problem is NP-hard [1], we are interested in approximation algorithms.

The interference scheduling problem consists of two correlated subproblems: the *power assignment* and the *coloring*. By far, most literature focuses on scheduling with *uniform power assignment*, in which all pairs send at the same power (see, e.g., [8, 12, 18]). In other studies, the *linear power assignment* is considered, in which the power level for a pair (u, v) is chosen proportional to the path loss $d(u, v)^\alpha$. In the example above, we have seen that choosing powers proportional to the square root of the path loss might be an interesting alternative. All these power assignments have the advantage that they are locally computable independent of other requests, which allows for an immediate implementation in a distributed setting. These are examples of *oblivious power assignments* which mean the power level assigned to a pair is defined as a function of the path loss (or the distance) between the nodes of a pair.

2.1.1 Outline

In Sect. 2.2, we formally introduce the physical model with SINR constraints and show a helpful robustness property of this model. In Sect. 2.3, we study scheduling algorithms for the linear power assignment. In particular, we introduce a *measure of interference* giving lower bounds for the schedule length with respect not only to linear but also to other power assignments. Based on this measure, we devise distributed scheduling algorithms for the linear power assignment achieving the minimal schedule length up to small factors. In Sect. 2.4, we study the square root power assignment. We show that this power assignment leads to better approximation guarantees in two kinds of problem instances defined by directed and bidirectional communication request. In Sect. 2.5, we study the limitations of oblivious power assignments by proving lower bounds for this approach. Finally, in Sect. 2.6 we summarize the results from our presentation with pointers to the literature and open problems.

2.2 Notation and Preliminaries

Let the path loss exponent $\alpha \geq 1$ and the gain $\beta > 1$ be fixed. Let V be a set of nodes from a metric space. Let $d(u, v)$ denote the distance between two nodes u and v. One is given a set R of n requests consisting of pairs $(u_i, v_i) \in V^2$, where u_i represents the source and v_i the destination of the signal from the ith request. W.l.o.g., we assume $\min_{i \in R} d(u_i, v_i) = 1$. Let $\Delta = \max_{i \in R} d(u_i, v_i)$ be the *aspect ratio*. We say that a set R of requests is a *nearly equilength* set, if the lengths of the requests in R differ by at most factor 2.

In the *interference scheduling problem* one needs to specify a power level $p_i > 0$ and a color $c_i \in [k] := \{1, \ldots, k\}$ for every $i \in [n] := \{1, \ldots, n\}$ such that the *latency*, i.e., the number of colors k, is minimized and the pairs in each color class satisfy the *SINR constraint*, that is, for every $i \in [n]$, it holds

$$\frac{p_i}{d(u_i, v_i)^\alpha} \geq \beta \left(\sum_{\substack{j \in [n] \setminus \{i\} \\ c_j = c_i}} \frac{p_j}{d(u_j, v_i)^\alpha} + \nu \right) \tag{2.1}$$

The SINR constraint is the central condition for successful communication in the physical model. It characterizes the received strength of the signal emitted from u_i at receiver v_i compared to *ambient noise* ν and the *interference* from signals of all other senders in the same color class. The so-called *scheduling complexity* of R, as introduced by Moscribroda and Wattenhofer [14], is the minimal number of colors (steps) needed to schedule all requests in R.

In this chapter we focus on distance-based power assignments because of their simplicity and locality, which is a striking conceptual advantage in distributed wireless systems. An *oblivious* (or distance-based) power assignment p is given by $p_i = \phi(d(u_i, v_i))$ with a function $\phi : [1, \Delta] \rightarrow (0, \infty)$. For uniqueness we assume that ϕ is always scaled such that $\phi(1) = 1$. Examples are the *uniform* $\phi(d(u_i, v_i)) = 1$ or the *linear* $\phi(d(u_i, v_i)) = d(u_i, v_i)^\alpha$ power assignment. Recently, the *square root* assignment $\phi(d(u_i, v_i)) = d(u_i, v_i)^{\alpha/2}$ has attracted some interest [5, 9] as it yields better approximation ratios for request scheduling than the uniform and the linear power assignment.

We define the *relative interference* on a request i from a request set R as

$$\mathrm{RI}_i(R) = c_i \cdot \frac{d(u_i, v_i)^\alpha}{p_i} \cdot \sum_{j \in R} \frac{p_j}{d(u_j, v_i)^\alpha}$$

where

$$c_i = \frac{\beta}{1 - \beta \cdot \nu \cdot p_i / d(u_i, v_i)}$$

denotes a constant that indicates the extent to which the ambient noise approaches the required signal at the receiver of request i. The relative interference describes the received interference at receiver v_i normalized by the received signal strength. The relative interference satisfies the two following properties for a request set R. First, R is SINR feasible iff for every $i \in R$, $\mathrm{RI}_i(R) \leq 1$. Second, the relative interference function is additive, that is, for every partition $R = R_1 \dot{\cup} R_2$ and every request i it holds $\mathrm{RI}_i(R) = \mathrm{RI}_i(R_1) + \mathrm{RI}_i(R_2)$.

We denote with an *r-signal* set or schedule one where each requests relative interference is at most $1/r$.

2.2.1 Robustness of the Physical Model

The main criticism of graph-based models is that they are too simplistic to model real wireless networks. The physical model requires simplifying assumptions, too, as (2.1) models no obstructions, perfectly isotropic radios and a constant ambient noise level.

In the following proof (from [10]) we show that there are only minor changes in the schedule length, if there are minor changes in the signal requirements. This justifies the analytic study of the physical model despite its simplifying assumptions.

Proposition 1 *Let R be a r-signal schedule under a power assignment p. Then there exists a r'-signal schedule R' for p that is at most $\lceil 2r'/r \rceil^2$ times longer than R, for $r' > r$.*

Proof Let R be a r-signal schedule and T be a single schedule step. We show that we can decompose T in at most $\lceil 2r'/r \rceil$ slots $T_1, T_2, \ldots$ that are r'-signal sets. We now process the requests in T by increasing index. For request i, assign it to the first set T_j, in which the relative interference on i is at most $1/2r'$. Since every request had at most a relative interference of $1/r$, it follows from the additivity of relative interference that there are at most

$$\left\lceil \frac{1/r}{1/2r'} \right\rceil = \left\lceil \frac{2r'}{r} \right\rceil$$

such sets. In each of these sets T_j the relative interference from requests with lower index is at most $1/2r'$. Now, for each of these sets we repeat this process, processing the requests in T_{ji}, now in reverse order. Using the same arguments T_j is split into at most $\lceil 2r'/r \rceil$ sets. In that way we make sure that the requests in each set have a relative interference of at most $1/2r'$ from requests with higher index, which bounds the total relative interference on each request by $1/r'$, while using at most $\lceil 2r'/r \rceil^2$ times more slots than the original schedule. □

2.3 Scheduling with the Linear Power Assignment

In the first part we focus on the linear power assignment, i.e., the power for a request pair (u_i, v_i) is equal to $d(u_i, v_i)^\alpha$ and, hence, linear in the path loss. The linear power assignment has the advantage of being energy efficient as the minimal transmission power required to transmit along a distance $d(u_i, v_i)$ is proportional to $d(u_i, v_i)^\alpha$.

We first present a measure of interference I, which allows us to lower bound the schedule for general metrics using the linear power assignment by $\Omega(I)$. If we allow any power assignment, the schedule length can be bounded by $\Omega(I/\log\Delta\log n)$. For $\alpha > 2$, embedding the instance in the Euclidean space improves this bound to $O(I/\log\Delta)$.

These results are complemented by a simple and efficient algorithm computing a schedule using $O(I \cdot \log n)$ steps. A more sophisticated algorithm computes a schedule using $O(I + \log^2 n)$ steps. This gives a constant factor approximation of the optimal schedule using the linear power assignment for dense instances, i.e., if $I \geq \log^2 n$.

2.3.1 Measure of Interference and Lower Bounds

We first present an instance-based *measure of interference* I, which allows us to lower bound the number of steps needed for scheduling a request set R in terms of I.

Definition 1 (Measure of Interference) Let $R \subseteq V \times V$ be a set of requests. For $w \in V$ define

$$I_w(R) = \sum_{(u,v)\in R} \min\left(1, \frac{d(u,v)^\alpha}{d(u,w)^\alpha}\right)$$

Using this function we define the measure of interference induced by the requests in R:

$$I = I(R) = \max_{w\in V} I_w(R)$$

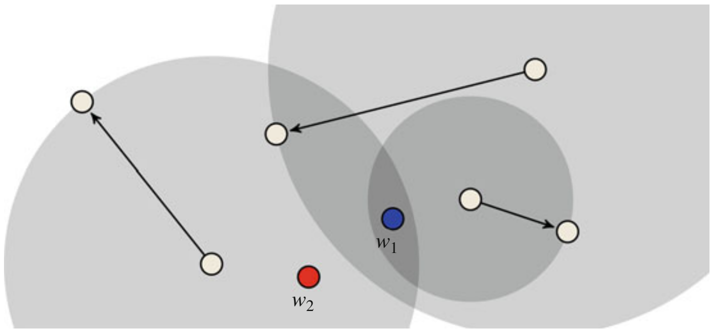


Fig. 2.2 An example for the measure of interference with three requests. *Gray* circles mark the areas where the interference from a sender is at least 1. For the *red* node I_{w_2} is 1 plus the interference from the two rightmost senders (each less than 1). The interference is maximal at the *blue* node w_1, i. e., $I_{w_1} = 3$, so the measure of interference I for this instance is $I = 3$

An example of the measure of interference is illustrated in Fig. 2.2.

Observe that I is subadditive, i.e., for $R = R_1 \cup R_2$ it holds

$$\begin{aligned} I(R) &= \max_{w \in V} I_w(R) \le \max_{w \in V} \{I_w(R_1) + I_w(R_2)\} \\ &\le \max_{w \in V} I_w(R_1) + \max_{w \in V} I_w(R_2) = I(R_1) + I(R_2) \end{aligned}$$

Theorem 1 *Let T be the minimum schedule length for a set of requests R with the linear power assignment. Then we have $T = \Omega(I)$.*

Proof Let there be a schedule of length T when using the linear power assignment. Then there exist sets of requests $R_1, \ldots, R_T$ each of which satisfies the SINR constraint for this power assignment. As I is subadditive we have $I\left(\bigcup_{t=1}^{T} R_t\right) \le \sum_{t=1}^{T} I(R_t)$. Thus it suffices to show that $I(R_t) = O(1)$ for every $t \in \{1, \ldots, T\}$, as this implies $T = \Omega(I)$.

Let $R_t = \{(u_1, v_1), \ldots, (u_{\bar{n}}, v_{\bar{n}})\}$ and let $w \in V$. Furthermore, let v_j be the receiver from R_t that is closest to w, i.e., $j \in \arg\min_{i \in [\bar{n}]} d(v_i, w)$. Possibly $w = v_j$.

We distinguish between two kinds of requests. We define a set U of indices of requests whose senders u_i lie within a distance of at most $\frac{1}{2} d(v_j, w)$ from w, i.e., $U = \{i \in [\bar{n}] \mid d(u_i, w) \le \frac{1}{2} d(v_j, w)\}$. Using the triangle inequality we can conclude for all $i \in U$:

$$d(u_i, v_j) \le d(u_i, w) + d(w, v_j) \le \frac{3}{2} d(v_j, w) \tag{2.2}$$

In addition, we have

$$d(v_j, w) \le d(v_i, w) \le d(v_i, u_i) + d(u_i, w) \le d(v_i, u_i) + \frac{1}{2} d(v_j, w)$$

Here the first equation holds since v_j is the closest receiver to w, the second equation holds by triangle inequality and the third step follows from the definition of U. This implies

$$d(v_j, w) \leq 2d(u_i, v_i) \tag{2.3}$$

Combining (2.2) and (2.3) we get $d(u_i, v_j) \leq 3d(u_i, v_i)$. Thus it holds

$$|U \setminus \{j\}| = \sum_{\substack{i \in U \\ i \neq j}} \frac{d(u_i, v_i)^\alpha}{d(u_i, v_i)^\alpha} \leq \sum_{\substack{i \in U \\ i \neq j}} \frac{d(u_i, v_i)^\alpha}{\frac{1}{3^\alpha} d(u_i, v_j)^\alpha} \leq \frac{3^\alpha}{\beta}$$

Hence,

$$I_w(U) = \sum_{i \in U} \min \left\{ 1, \frac{d(u_i, v_i)^\alpha}{d(u_i, w)^\alpha} \right\} \leq \frac{3^\alpha}{\beta} + 1$$

Next we upper bound $I_w(R_t \setminus U)$. For all $i \in [\bar{n}] \setminus U$ it holds that

$$d(u_i, v_j) \leq d(u_i, w) + d(w, v_j) \leq d(u_i, w) + 2d(u_i, w) = 3d(u_i, w)$$

by applying triangle inequality and the definition of U. As a consequence

$$I_w(R_t \setminus U) \leq \sum_{\substack{i \in [\bar{n}] \setminus U \\ i \neq j}} \frac{d(u_i, v_i)^\alpha}{d(u_i, w)^\alpha} \leq \sum_{\substack{i \in [\bar{n}] \setminus U \\ i \neq j}} \frac{d(u_i, v_i)^\alpha}{\frac{1}{3^\alpha} d(u_i, v_j)^\alpha} \leq \frac{3^\alpha}{\beta}$$

Thus

$$I_w(R_t) \leq I_w(U) + I_w(R_t \setminus U) = \frac{2 \cdot 3^\alpha}{\beta} + 1 = O(1)$$

□

Theorem 2 *Let T denote the optimal schedule length using any power assignment. Then we have $T = \Omega\,(I / \log \Delta \cdot \log n)$.*

Proof We use a similar technique as in the proof of Theorem 1. However, we have to deal with an unknown power assignment. Since there is a schedule of length T in this power assignment, there exist sets of requests $R_1, \ldots, R_T$ each of which satisfies the SINR constraint for this power assignment. We divide such a set R_t into $\log \Delta$ classes $C_{t,j} = \{(u, v) \in R_t \mid 2^{j-1} \leq d(u, v) < 2^j\}$. Again, by using the subadditivity of I, it suffices to show that $I(C_{t,j}) = O(\log n)$ for such a class. Fix $C_{t,j}$ and let $C_{t,j} = \{(u_1, v_1), \ldots, (u_{\bar{n}}, v_{\bar{n}})\}$. Further, for notational simplicity we write $L = 2^{j-1}$.

As an important fact we can bound the number of requests whose senders are located around a node within a distance of at most ℓ.

Lemma 1 *For all $w \in V$, $\ell \geq L$ we have for $K_\ell(w) = \{i \in [\bar{n}] \mid d(u_i, w) \leq \ell\}$:*

$$|K_\ell(w)| \leq \frac{1}{\beta}\left(\frac{4\ell}{L}\right)^\alpha + 1$$

Proof Let p be the power assignment that allows all requests to be served in a single time slot. Let furthermore (u_k, v_k) be the request with $k \in K_L(w)$ that is transmitted with minimal power p_k. As the SINR condition is satisfied for request (u_k, v_k), we get

$$\frac{1}{\beta}\frac{p_k}{d(u_k, v_k)^\alpha} \geq \sum_{\substack{i \in K_\ell(w)\\ i \neq k}} \frac{p_i}{d(u_i, v_k)^\alpha} \geq \sum_{\substack{i \in K_\ell(w)\\ i \neq k}} \frac{p_i}{(2\ell + 2L)^\alpha} \geq \frac{(|K_\ell(w)| - 1) \cdot p_k}{(2\ell + 2L)^\alpha}$$

So

$$|K_\ell(w)| - 1 \leq \frac{1}{\beta}\left(\frac{2\ell + 2L}{d(u_k, v_k)}\right)^\alpha \leq \frac{1}{\beta}\left(\frac{4\ell}{L}\right)^\alpha$$

□

Now, let $w \in V$. We prove $I_w(C_{t,j}) = O(\log n)$. W. l. o. g., let $u_1, \ldots, u_{\bar{n}}$ be ordered by increasing distance to w. Note that for all $\ell > 0$ we have $K_\ell(w) = \{1, \ldots, x\}$ for some $x \in \mathbb{N}$ by this definition.

For $k \leq \log \bar{n} + 1$ let $S_k = [2^k] \setminus [2^{k-1}]$. Furthermore, let ℓ_k be defined as $\ell_k = \min_{i \in S_k} d(u_i, w)$. For the value of $I_w(C_{t,j})$ follows from these definitions:

$$\begin{aligned} I_w(C_{t,j}) &= \sum_{i=1}^{\bar{n}} \min\left\{1, \frac{d(u_i, v_i)^\alpha}{d(u_i, w)^\alpha}\right\} \\ &\leq \sum_{k=1}^{\log \bar{n}+1} \sum_{i \in S_k} \frac{d(u_i, v_i)^\alpha}{d(u_i, w)^\alpha} + \sum_{i \in K_L(w)} 1 \leq (2L)^\alpha \sum_{k=1}^{\log \bar{n}+1} \frac{|S_k|}{\ell_k^\alpha} + |K_L(w)| \end{aligned}$$

As the distances are increasing, we have $\ell_k \geq d(u_i, w)$ for all $i \leq 2^{k-1}$. In other words $[2^{k-1}] \subseteq K_{\ell_k}(w)$.

Since we add up the interference induced by requests from $K_L(w)$ separately, we may assume $\ell_k \geq L$ for all k and thus apply Lemma 1 on $|K_{\ell_k}(w)|$, which gives

$$2^{k-1} = |[2^{k-1}]| \leq |K_{\ell_k}(w)| \leq \left(\frac{4\ell_k}{L}\right)^\alpha + 1$$

Consequently, we have

$$\ell_k^\alpha \geq (2^{k-1} - 1)\left(\frac{L}{4}\right)^\alpha$$

Using the above results for ℓ_k^α and $|K_L(w)|$ we can bound $I_w(C_{t,j})$ by

$$(2L)^\alpha \sum_{k=1}^{\log \bar{n}+1} \frac{2^{k-1}}{(2^{k-1}-1)\left(\frac{L}{4}\right)^\alpha} + \left(\frac{4^\alpha}{\beta} + 1\right) \leq 8^\alpha \sum_{k=1}^{\log \bar{n}+1} 2 + \frac{4^\alpha}{\beta} + 1 = O(\log n)$$

□

Earlier results restricted the instances often to the Euclidean plane and required α to be strictly greater than 2. Under these assumptions we can use geometric arguments to get an even better bound of $\Omega(I/\log \Delta)$ on the optimal schedule length, as we show in the following.

Theorem 3 *Let the instance be located in the Euclidean plane, let $\alpha > 2$, and let T denote the optimal schedule length using any power assignment. Then we have $T = \Omega\,(I/\log \Delta)$.*

Proof Again, we divide the requests into $\log \Delta \cdot T$ classes $C_{t,i}$. This time, we have to prove $I_w(C_{t,i}) = O(1)$. Let us remark that in the Euclidean plane a ring of inner radius $L \cdot r$ and width L can be covered by $8(r+1)$ circles of radius L. If x is the center of such a circle, we get from Lemma 1 that $|K_L(x)| \leq \frac{4^\alpha}{\beta}$. Thus we have $|K_{L(r+1)}(w) \setminus K_{Lr}(w)| \leq 8(r+1)\frac{4^\alpha}{\beta} \leq 16r\frac{4^\alpha}{\beta} = r\frac{4^{\alpha+2}}{\beta}$ for $r \geq 1$. We can bound $I_w(C_{t,j})$ by

$$I_w(C_{t,j}) \leq \sum_{r=1}^{\infty} |K_{L(r+1)}(w) \setminus K_{Lr}(w)| \cdot \frac{(2L)^\alpha}{(Lr)^\alpha} + |K_L(w)|$$

Using the above result we get

$$I_w(C_{t,j}) \leq 2^\alpha \frac{4^{\alpha+2}}{\beta} \sum_{r=1}^{\infty} r^{1-\alpha} + \frac{4^\alpha}{\beta} \leq \frac{4^\alpha}{\beta}\left(2^\alpha 4^2 \frac{\alpha-1}{\alpha-2} + 1\right) = O(1)$$

□

2.3.2 Upper Bounds for the Linear Power Assignment

The measure of interference enables us to design randomized algorithms using the linear power assignment, i.e., the power for the transmission from u to v is $c \cdot d(u,v)^\alpha$ for some fixed $c \geq \beta\nu$. As a key fact, we can simplify the SINR constraint in this setting as follows. If R is a set of requests that can be scheduled in one time slot, we have for all nodes v' with $(u', v') \in R$

$$\sum_{\substack{(u,v)\in R\\(u,v)\neq(u',v')}} \frac{c \cdot d(u,v)^{\alpha}}{d(u,v')^{\alpha}} \leq \frac{c}{\beta} - \nu$$

Since $\beta > 1$ we can write equivalently

$$I_{v'}(R) = \sum_{(u,v)\in R} \min\left\{1, \frac{d(u,v)^{\alpha}}{d(u,v')^{\alpha}}\right\} \leq \frac{1}{\beta} - \frac{\nu}{c} \tag{2.4}$$

For simplicity of notation we replace $\frac{1}{\beta} - \frac{\nu}{c}$ by $\frac{1}{\beta'}$ in the following proofs.

The idea of our basic algorithm (Algorithm 1) is that each sender decides randomly in each time slot if it tries to transmit until it is successful. The probability of transmission is set to $\frac{1}{2\beta' I}$ and is not changed throughout the process.

Algorithm 1 A simple single-hop algorithm

```
1: while packet has not been successfully transmitted do
2:     try transmitting with probability $\frac{1}{2\beta' I}$
3: end while
```

Theorem 4 *Algorithm 1 generates a schedule of length at most $O(I \log n)$ whp.*

Proof Let us first consider the probability of success for a fixed request (u_k, v_k) in a single step of the algorithm. Let X_i, $i \in [n]$, be the 0/1 random variable indicating if sender u_i tries to transmit in this step. Assume a sender u_k tries to transmit in this step, i.e., $X_k = 1$. To make this attempt successful, the interference constraint (2.4) has to be satisfied. We can express this event as $Z \leq 1/\beta'$ where Z is defined by

$$Z = \sum_{\substack{i\in[n]\\i\neq k}} \min\left\{1, \frac{d(u_i,v_i)^{\alpha}}{d(u_i,v_k)^{\alpha}}\right\} X_i$$

We have $\mathbf{E}[Z] \leq 1/2\beta'$ and thus we can use Markov's inequality to bound the probability that this packet cannot be transmitted successfully by

$$\mathbf{Pr} Z \geq \frac{1}{\beta'} \leq \mathbf{Pr} Z \geq 2\mathbf{E}[Z] \leq \frac{1}{2}$$

To make the transmission successful the two events $X_k = 1$ and $Z \leq 1/\beta' \frac{1}{\beta'}$ have to occur. Since they are independent it holds that

$$\mathbf{Pr} X_k = 1, Z \leq \frac{1}{\beta'} = \mathbf{Pr} X_k = 1 \cdot \mathbf{Pr} Z \leq \frac{1}{\beta'} \geq \frac{1}{2\beta' I}\left(1 - \frac{1}{2}\right) = \frac{1}{4\beta' I}$$

The probability for packet k not to be successfully transmitted in $(k_0 + 1)4\beta' I \ln n$ independent repeats of such a step is therefore at most

$$\left(1 - \frac{1}{4\beta' I}\right)^{(k_0+1)4\beta' I \ln n} \leq e^{-(k_0+1)\ln n} = n^{-(k_0+1)}$$

Applying a union bound we get an overall bound on the probability that one of n packets is not successfully transmitted in these independent repeats by n^{-k_0}. This means all senders are successful within $O(I \log n)$ steps whp.

An obvious disadvantage of the basic algorithm is that the probability of transmission stays the same throughout the process. To improve it, one idea could be to increase the probability of transmission after some transmissions have successfully taken place. This is why we need the following weighted Chernoff bound that can deal with dependent random variables.

Lemma 2 *Let $X_1, \ldots, X_n$ be 0/1 random variables for which there is a $p \in [0, 1]$ such that for all $k \in [n]$ and all $a_1, \ldots, a_{k-1} \in \{0, 1\}$*

$$\mathbf{Pr} X_k = 1 \mid X_1 = a_1, \ldots X_{k-1} = a_{k-1} \leq p \tag{2.5}$$

Let furthermore $w_1, \ldots, w_n$ be reals in $(0, 1]$ and $\mu \geq p \sum w_i$. Then the weighted Chernoff bound

$$\mathbf{Pr} \sum_{i=1}^{n} w_i X_i \geq (1+\delta)\mu \leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu}$$

holds.

Proof (*Sketch*). To show this bound, a standard proof for the weighted Chernoff bound [16] can be adapted. By using the definition of expectation and repeatedly applying (2.5), one can show that

$$\mathbf{E}\left[e^{tX}\right] \leq \prod_{i=1}^{n} \left(p e^{t w_i} + 1 - p\right)$$

although random variables are no more independent. In the original proof no other step makes use of the independence. □

We can now use this bound to analyze Algorithm 2. This algorithm assigns random delays to all packets. The maximum delay is decreased depending on I^{curr}, which denotes the measure of interference that is induced by the requests that have not been scheduled at this point.

The algorithm works as follows: During one iteration of the outer *while* loop by repeatedly assigning random delays to the packets the measure of interference is

Algorithm 2 An $O(I + \log^2 n)$ whp algorithm

```
1: while $I^{\text{curr}} \geq \log n$ do
2:     $J := I^{\text{curr}}$
3:     while $I^{\text{curr}} \geq \frac{J}{2}$ do
4:         if packet $i$ has not been successfully transmitted then
5:             assign a delay $1 \leq \delta_i \leq 16e\beta' J$ i. u. r.
6:             try transmission after waiting the delay
7:         end if
8:     end while
9: end while
10: execute algorithm Algorithm 1
```

reduced to a half of its initial value. This is repeated until we have $I^{\text{curr}} < \log n$ and the basic algorithm is applied.

Our first observation is that reducing I^{curr} by factor 2 takes $O(I^{\text{curr}})$ scheduling steps whp.

Lemma 3 *During one iteration of the outer* while *loop of Algorithm 2, the inner* while *loop is executed at most $k_0 + 2$ times with probability at least $1 - n^{-k_0}$ for all constants k_0.*

Proof Let us first consider a single iteration of this loop. We assume all senders are taking part as if none has been successful during this iteration of the outer *while* loop yet. We only benefit from any previous success.

Observe, if the senders of a set S are transmitting and there is a collision for packet i we have

$$\sum_{\substack{j \in S \\ j<i}} \min\left\{1, \frac{d(u_j, v_j)^\alpha}{d(u_j, v_i)^\alpha}\right\} > \frac{1}{2\beta'} \quad \text{or} \quad \sum_{\substack{j \in S \\ j>i}} \min\left\{1, \frac{d(u_j, v_j)^\alpha}{d(u_j, v_i)^\alpha}\right\} > \frac{1}{2\beta'}$$

In the first case let $Y_i^< = 1$, in the second one $Y_i^> = 1$. We now show that the random variables $Y_1^<, \ldots, Y_n^<$ fulfill (2.5) for $p = \frac{1}{8e}$. Let us fix $k \in [n]$ and $a_1, \ldots, a_{k-1} \in \{0, 1\}$. We have to show $\mathbf{Pr} Y_k^< = 1 \mid Y_1^< = a_1, \ldots, Y_{k-1}^< = a_{k-1} \leq p$.

Since the delays δ_i are drawn independently they can be considered as if they were drawn one after the other in the order $\delta_1, \delta_2, \ldots$. Then the value of $Y_i^<$ would already be determined after drawing δ_i by definition. In other words, the values of $\delta_1, \ldots, \delta_{k-1}$ already determine the values of $Y_1^<, \ldots, Y_{k-1}^<$. It follows that there is a subset $M \subseteq [16e\beta' J]^{k-1}$ of delay values such that $Y_1^< = a_1, \ldots, Y_{k-1}^< = a_{k-1}$ iff $(\delta_1, \ldots, \delta_{k-1}) \in M$.

Now let X_i be a 0/1 random variable for $i \in [k-1]$ such that $X_i = 1$ iff $\delta_i = \delta_k$. We can observe that we have for all $(b_1, \ldots, b_{k-1}) \in [16e\beta' J]^{k-1}$:

$$\mathbf{E}\left[X_i \mid \delta_1 = b_1, \ldots, \delta_{k-1} = b_{k-1}\right] = \frac{1}{16e\beta' J}$$

Define furthermore

$$Z_k^< = \sum_{i=1}^{k-1} \min\left\{1, \frac{d(u_i, v_i)^\alpha}{d(u_i, v_k)^\alpha}\right\} X_i$$

with $\mathbf{E}\left[Z_k^< \mid \delta_1 = b_1, \dots, \delta_{k-1} = b_{k-1}\right] \le \frac{1}{16e\beta'}$. Now it holds that

$$\begin{aligned}
&\mathbf{Pr}\,[\ Y_k^< = 1 \ \mid\ \delta_1 = b_1, \dots, \delta_{j-1} = b_{k-1}\ ]\\
&\qquad = \mathbf{Pr} Z_k^< > \frac{1}{2\beta'} \,\middle|\, \delta_1 = b_1, \dots, \delta_{k-1} = b_{k-1}\\
&\qquad \le 2\beta' \mathbf{E}\left[Z_k^< \mid \delta_1 = b_1, \dots, \delta_{k-1} = b_{k-1}\right]\\
&\qquad = \frac{1}{8e} = p
\end{aligned}$$

We now apply the law of alternatives:

$$\begin{aligned}
&\mathbf{Pr} Y_k^< = 1 \mid Y_1^< = a_1, \dots, Y_{k-1}^< = a_{k-1}\\
=& \sum_{(b_1,\dots,b_{k-1})\in M} \mathbf{Pr}\delta_1 = b_1, \dots, \delta_{k-1} = b_{k-1} \mid Y_1^< = a_1, \dots, Y_{k-1}^< = a_{k-1}\\
&\quad \cdot \mathbf{Pr} Y_k^< = 1 \mid \delta_1 = b_1, \dots, \delta_{k-1} = b_{k-1}\\
\le& \ p
\end{aligned}$$

Thus, for $w \in V$, we may apply Lemma 2 on $I_w^<$ defined as follows:

$$I_w^< = \sum_{i=1}^{n} \min\left\{1, \frac{d(u_i, v_i)^\alpha}{d(u_i, w)^\alpha}\right\} Y_i^<$$

This random variable indicates the remaining measure of interference that is caused by these collisions. Setting $\delta = 2e - 1$ and $\mu = \frac{J}{8e}$ Lemma 2 states

$$\mathbf{Pr} I_w^< \ge \frac{J}{4} \le 2^{-\frac{J}{4}} \le n^{-1}$$

Now consider the situation after $k_0 + 2$ iterations of the inner *while* loop. Since these are independent repeats we have

$$\mathbf{Pr} I_w^< \ge \frac{J}{4} \le n^{-(k_0+2)}$$

With a symmetric argument this also applies to $I_j^>$. For a sender that has not been successful we have $Z_j^< + Z_j^> \ge 1$. This means we have the bound $I_w^{\text{curr}} \le I_w^< + I_w^>$. For the remaining measure of interference $I^{\text{curr}} = \max_{w\in V} I_w^{\text{curr}}$ we can conclude

$$\begin{aligned}
\mathbf{Pr} I^{\text{curr}} \geq \frac{J}{2} &\leq \sum_{w \in V} \mathbf{Pr} I_w^{\text{curr}} \geq \frac{J}{2} \\
&\leq \sum_{w \in V} \mathbf{Pr} I_w^{<} \geq \frac{J}{4} \text{ or } I_w^{<} \geq \frac{J}{4} \\
&\leq n \left(n^{-(k_0+2)} + n^{-(k_0+2)} \right) \\
&\leq n^{-k_0}
\end{aligned}$$

□

Using the previous lemma, we can bound the numbers of steps that are generated in the *while* loops.

Theorem 5 *Algorithm 2 generates a schedule of length at most $O(I + \log^2 n)$ steps whp.*

Proof Let T_k denote the number of scheduling steps generated in the kth execution of the outer *while* loop. As shown in the previous lemma, it holds that

$$\mathbf{Pr} v_k \geq (k_0 + 3)\, 16 e \beta' \frac{1}{2^{k-1}} I \leq \frac{1}{n^{k_0+1}}$$

Let furthermore U denote the number of scheduling steps generated in the execution of Algorithm 1. As shown in Lemma 4, it holds that

$$\mathbf{Pr} U \geq (k_0 + 2)\, 4 \beta' \ln n \log n \leq \frac{1}{n^{k_0+1}}$$

Thus the total number of steps generated in the *while* loops $\sum_k v_k + U$ can be estimated by

$$\begin{aligned}
&\mathbf{Pr} \sum_k v_k + U \geq (k_0 + 3)\, 32 e \beta' I + (k_0 + 2)\, 4 \beta' \ln n \log n \\
&\leq \mathbf{Pr} \bigvee_k v_k \geq (k_0 + 3)\, 16 e \beta' \frac{1}{2^{k-1}} I \vee U \geq (k_0 + 2)\, 4 \beta' \ln n \log n \\
&\leq \sum_k \mathbf{Pr} v_k \geq (k_0 + 3)\, 16 e \beta' \frac{1}{2^{k-1}} I + \mathbf{Pr} U \geq (k_0 + 2)\, 4 \beta' \ln n \log n \\
&\leq \sum_k \frac{1}{n^{k_0+1}} + \frac{1}{n^{k_0+1}} \\
&\leq (\log n + 1) \frac{1}{n^{k_0+1}} \\
&\leq \frac{1}{n^{k_0}}
\end{aligned}$$

This means the total number of steps upper bounded by

$$(k_0+3)\,32e\beta' I + (k_0+2)\,4\beta' \ln n \log n = O(I + \log^2 n)$$

with probability at least $1 - \frac{1}{n^{k_0}}$. □

In sufficiently dense instances, i.e., $I \geq \log^2 n$, this algorithm yields a constant-factor approximation for the optimal schedule compared to the linear power assignment with high probability. Compared to the optimal power assignment the approximation factor then is $O(\log \Delta \cdot \log n)$ whp for general metrics, respectively. $O(\log \Delta)$ for the two-dimensional Euclidean plane.

Algorithm 1 can be implemented in a distributed way losing a factor $\log n$ in the following way. In contrast to the centralized problem, the nodes do not know the correct value of I, thus, they do not know their transmission probability. Now in the distributed setting the algorithm processes in each *while* iteration $\log n$ steps, where in each of these steps the transmission probability is halfed, that is, starting by $1/2\beta'$ down to $1/2\beta' n$.

Algorithm 2 can be modified analogously, leading to a schedule of length $O(\log n \cdot (I + \log^2 n))$ whp.

2.4 Scheduling with the Square Root Power Assignment

The scheduling algorithms for the linear power assignment presented in Sect. 2.3 achieve an approximation factor of order $\log \Delta \text{polylog}\, n$ in comparison to an optimal solution with respect to general power assignments. In this section, we show that the square root power assignment admits schedules beating this bound achieving an approximation factor of order $\log\log \Delta \text{polylog}\, n$. Furthermore, we present a bidirectional variant of the interference scheduling problem in which the square root power assignment yields an approximation of order polylog n and is, hence, independent of the aspect ratio.

2.4.1 Scheduling Directed Requests

In this section we show how to achieve an $O(\log\log \Delta \log^3 n)$ approximation on the interference scheduling problem using square root power. To prove this result we first show two properties that make use of the following definitions. We call a set R of requests *well separated*, if the length of any pair of requests differs by a factor of either at most 2 or at least $16n^{2/\alpha}$. We say that two requests (u_i, v_i) and (u_j, v_j) are *τ-close* under the square root power assignment if $\max\{\text{RI}_i(j), \text{RI}_j(i)\} \geq \tau$.

Lemma 4 *Let R be a well-separated SINR-feasible set of requests. Let (u_0, v_0) be a request that is shorter than the requests in R by at least a factor of $16n^{2/\alpha}$. If all the requests in R are $1/2n$-close to (u_0, v_0) under the square root power assignment, then $|R| = O(\log\log \Delta)$.*

Proof Let R' be a maximum 3^α-signal subset of R, let n' denote the number of requests in R' and w.l.o.g. let the requests in R' be labeled in increasing order of length. From Proposition 1 we know $\left|R'\right| = n' \geq |R|/9^\alpha$. As all the requests in R' are $1/2n$-close to (u_0, v_0), R' consists of two types of requests:

- Requests j for which the ratio between j's interference and the received signal from u_0 at receiver v_0 is at least $1/2n$ (or $\sqrt{d(u_0, v_0) \cdot d(u_j, v_j)}^\alpha \geq d(u_j, v_0)^\alpha \cdot \frac{1}{2n}$) and
- Requests j for which the ratio between u_0's interference and the received signal from j's sender at v_j is at least $1/2n$ (or $\sqrt{d(u_0, v_0) \cdot d(u_j, v_j)}^\alpha \geq d(u_0, v_j)^\alpha \cdot \frac{1}{2n}$).

We only consider the former type, the argument is almost identical for the latter type and will be left to the reader.

Let $j, j' \in R'$, w.l.o.g. assume $j \geq j'$. As they are $1/2n$-close to (u_0, v_0), it holds $\sqrt{d(u_0, v_0) \cdot d(u_j, v_j)}^\alpha \geq d(u_j, v_0)^\alpha \cdot \frac{1}{2n}$ (and analogously for j'). So we get

$$d(u_j, v_0) \leq \sqrt{d(u_0, v_0) \cdot d(u_j, v_j)}(2n)^{1/\alpha}$$

and

$$d(u_{j'}, v_0) \leq \sqrt{d(u_0, v_0) \cdot d(u_{j'}, v_{j'})}(2n)^{1/\alpha}$$

With triangle inequality we can conclude

$$\begin{aligned} d(u_{j'}, v_j) &\leq d(u_{j'}, v_i) + d(v_i, u_j) + d(u_j, v_j) \\ &\leq d(u_j, v_j) + 2^{1+1/\alpha} n^{1/\alpha} \sqrt{d(u_0, v_0) \cdot d(u_j, v_j)} \end{aligned}$$

Applying $\alpha \geq 1$ and $d(u_j, v_j) \geq 16n^{2/\alpha} d(u_0, v_0)$ to this inequality, we get

$$d(u_{j'}, v_j) \leq d(u_j, v_j) + 2^{1+1/\alpha} n^{1/\alpha} \sqrt{d(u_0, v_0) \cdot d(u_j, v_j)} \leq 2d(u_j, v_j)$$

For technical simplicity, we use the more relaxed $d(u_{j'}, v_j) < 3d(u_j, v_j)$ in the following. Using the same arguments as above we get

$$d(u_j, v_{j'}) \leq d(u_{j'}, v_{j'}) + 2^{1+1/\alpha} n^{1/\alpha} \sqrt{d(u_0, v_0) \cdot d(u_j, v_j)}$$

Multiplying this inequality with $d(u_{j'}, v_j) < 3d(u_j, v_j)$ it follows

$$(u_{j'}, v_j) \cdot d(u_j, v_{j'}) < 3d(u_j, v_j) d(u_{j'}, v_{j'}) + 12n^{1/\alpha} d(u_j, v_j) \sqrt{d(u_0, v_0) \cdot d(u_j, v_j)}$$

Since R' is a 3^α-signal set, we have $d(u_{j'}, v_j) \cdot d(u_j, v_{j'}) \geq 9d(u_j, v_j) \cdot d(u_{j'}, v_{j'})$. Again, applying the well separation, the last two inequalities yield (with canceling a $6d(u_j, v_j)$ factor)

$$d(u_{j'}, v_{j'}) < 2n^{1/\alpha}\sqrt{d(u_0, v_0) \cdot d(u_j, v_j)} \tag{2.6}$$

This equation implies $d(u_j, v_j) > 2d(u_{j'}, v_{j'})$. By well separation of R it follows $d(u_j, v_j) \geq 16n^{2/\alpha}d(u_{j'}, v_{j'})$. Now it follows from (2.6)

$$d(u_{i+1}, v_{i+1}) \geq \frac{d(u_i, v_i)^2}{4d(u_0, v_0)n^{2/\alpha}} \geq \frac{2d(u_i, v_i)^2}{d_1}$$

for any $i \in \{2, \ldots, n'\}$. Let $\lambda_i = d(u_i, v_i)/d(u_1, v_1)$. Then $\lambda_{i+1} \geq 2\lambda_i^2$ and by induction $\lambda_{n'} \geq 2^{2^{n'-1}-1}$. Hence, $n' = \left|R'\right| \leq \lg\lg\lambda_{n'} + 2 = \lg\lg\Delta + 2$, which proves the lemma. □

Lemma 5 *Let R be a well-separated set of requests. If any subset of R containing only nearly equilength requests can be scheduled with the linear power assignment using at most c colors, then all requests in R can be scheduled with $O(c \log\log \Delta)$ colors using the square root power assignment.*

Proof In the following we show that a single step from a schedule of R can be scheduled in $O(\log\log\Delta)$ steps. Let $R = R_1 \dot\cup R_2 \dot\cup \ldots \dot\cup R_t$ denote the decomposition of R in length groups, such that the length of the requests in each group differs by at most factor 2 and in different groups by at least factor $16n^{2/\alpha}$. First we transform the schedules for each length group in an r-signal schedule, with $r = 2^{\alpha/2}$. This changes the number of schedule steps by at most factor $(r+1)^2$ (by Proposition 1). Let $T = \bigcup_i T_i$ be a single schedule step from the schedule of R and let T_i denote the requests in T from length group R_i. W.l.o.g., let the requests in T be ordered by decreasing length.

Lemma 4 states that for each request i there are at most $O(\log\log\Delta)$ longer requests in T that are $1/2n$-close to i. Let $p = O(\log\log\Delta)$ denote this bound. Now process the requests $i \in T$ by decreasing length: Assign i to a step T'_j with $j \in [p+1]$ that does not contain a $1/2n$-close request for i.

It remains to show that this assignment yields a feasible schedule. Consider a request $i \in T'_j$ that originally came from set R_k. The relative interference on i from nearly equilength requests in $T'_j \cap R_k$ under the linear power assignment is at most $1/r$, since each length group is an r-signal set. We first analyze the influence from changing the power assignment from linear to square root in a length class. It holds for two requests a and b for the linear power assignment

$$\mathrm{RI}_a\left((u_b, v_b)\right) = c_a \cdot \frac{d(u_a, v_a)^\alpha}{p_a} \cdot \frac{p_b}{d(u_b, v_a)^\alpha} = c_a \cdot \frac{d(u_b, v_b)^\alpha}{d(u_b, v_a)^\alpha}$$

and for the square root power assignment

$$\mathrm{RI}_a\left((u_b, v_b)\right) = c_a \cdot \sqrt{d(u_a, v_a)^\alpha} \cdot \frac{\sqrt{d(u_b, v_b)}^\alpha}{d(u_b, v_a)^\alpha}$$

Since the requests in the same length class differ by at most factor 2 combining these two bounds yields that changing the power in a feasible schedule from the linear power assignment to the square root power assignment changes the relative interference by a factor of at most $2^{\alpha/2}$ in such nearly equilength request sets. Thus, the relative interference on i from requests in the same length class is at most $1/2$. On the other hand, the relative interference on i from requests not in the same length class is at most $1/2n$ each, by construction, which is at most $1/2$ in total. The relative interference on each link is not greater than one, which gives us an SINR-feasible schedule. □

Theorem 6 *Suppose there exists a ρ-approximate algorithm for the interference scheduling problem on nearly equilength request sets using uniform power assignment. Then there exists an $O(\rho \cdot \log\log \Delta \cdot \log n)$-approximate algorithm for the interference scheduling problem using the square root power assignment.*

Proof Let R be the set of requests. We partition R into $k = \left\lceil \frac{2}{\alpha} \log 16n \right\rceil$ well-separated sets as follows. Let $R_1, R_2, \ldots$ denote length groups with $R_i = \{j \in R \mid d(u_i, v_i) \in [2^{i-1}, 2^i)\}$. Then, partition R into classes $B_i = \cup_j R_{i+j\cdot k}$, for $i = 1, 2, \ldots, k$. Now the theorem follows from applying Lemma 5 on each of the classes B_i separately. □

Recall that Algorithm 2 had an approximation ratio of $O(\log \Delta \log^2 n)$ in general metrics. For nearly equilength request sets this ratio reduces to $O(\log^2 n)$, which gives the following result.

Corollary 1 *The interference scheduling problem in general metrics has an approximation factor of $O(\log\log \Delta \cdot \log^3 n)$ for the square root power assignment.*

For instances embedded in the Euclidean plane the approximation factor of Algorithm 2 is $O(\log \Delta \log n)$ which reduces to $O(\log n)$ for nearly equilength request sets.

Corollary 2 *For $\alpha > 2$, the interference scheduling problem in the two-dimensional Euclidean space has an approximation factor of $O(\log\log \Delta \cdot \log^2 n)$ for the square root power assignment.*

2.4.2 Scheduling Bidirectional Requests

Most communication protocols used in practice rely on bidirectional point-to-point communication. This is reflected by the following variant of the physical model in which requests are undirected, that is, each of the two nodes of a request acts as both sender and receiver. The SINR constraint is adapted as follows. For every request pair $(u_i, v_i) \in R$ and $w \in \{u_i, v_i\}$, it must hold

$$\frac{p_i}{d(u_i, v_i)^\alpha} \geq \beta \left(\sum_{\substack{j \in [n] \setminus \{i\} \\ c_j = c_i}} \frac{p_j}{\min\{d(u_j, w)^\alpha, d(v_j, w)^\alpha\}} + \nu \right)$$

In every request set that fulfills this condition the two nodes of a request can exchange messages in both directions, as long as only one of them acts as sender at any given time.

In this setting, bounded, linear, and superlinear power functions still can have schedule lengths of $\Omega(n)$, while the optimal schedule has constant length. This can be shown by a straightforward adaption of the proof for Theorem 8. For sublinear assignments this adaption is not possible. In fact, we show in the following that the square root power assignment guarantees an approximation factor of $O(\log^3 n)$.

First, we need the following technical lemma.

Lemma 6 *Let (u_i, v_i) and (u_j, v_j) be two requests. If they can be scheduled simultaneously, then*

$$\min\{d(w_i, w_j)\}^2 \geq \beta^{2/\alpha} \cdot d(u_i, v_i) \cdot d(u_j, v_j)$$

Proof Let $w_1 \in \{u_i, v_i\}$ and $w_2 \in \{u_j, v_j\}$, such that $\min\{d(w_i, w_j)\} = d(w_1, w_2)$. The SINR constraint gives

$$\frac{p_i}{d(u_i, v_i)^\alpha} \leq \beta \frac{p_j}{d(w_1, w_2)^\alpha}$$

and

$$\frac{p_j}{d(u_j, v_j)^\alpha} \leq \beta \frac{p_i}{d(w_1, w_2)^\alpha}$$

From multiplying both equations follows

$$d(w_1, w_2)^2 \geq \beta^{2/\alpha} \cdot d(u_i, v_i) \cdot d(u_j, v_j)$$

□

Lemma 7 *Let R be a set of requests that can be scheduled with an arbitrary power assignment and let i be a request. Then there is at most a constant number of requests $j \in R$ with $d(u_j, v_j) \geq n^{2/\alpha} \cdot d(u_i, v_i)$ that cause a relative interference of at least $1/2n$ on i under the square root power assignment.*

Proof In the following we show that for fixed β there is at most one request $j \in R$ with $d(u_j, v_j) \geq n^{2/\alpha} \cdot d(u_i, v_i)$ that causes a relative interference of at least $1/2n$ on i under the square root power assignment. By Proposition 1 this yields the claimed result.

Assume that there are two requests $j, j' \in R$ with $d(u_j, v_j)$ and $d(u_{j'}, v_{j'})$ at least $n^{2/\alpha} \cdot d(u_i, v_i)$ that cause a relative interference of more than $1/2n$ on i under the square root power assignment. W.l.o.g, let $d(u_j, v_j) \geq d(u_{j'}, v_{j'})$. For $k \in \{j, j'\}$ and $w \in \{u_i, v_i\}$ let $d_m = \min\{d(u_k, w), d(v_k, w)\}$. The relative interference under the square root power assignment implies

$$\left(\frac{\sqrt{d(u_k, v_k) d(u_i, v_i)}}{d_m}\right)^{\alpha} \geq \frac{1}{2n}$$

This implies

$$d_m \leq (2n)^{1/\alpha} \sqrt{d(u_k, v_k) \cdot d(u_i, v_i)} \leq (2n)^{1/\alpha} \sqrt{d(u_j, v_j) \cdot d(u_i, v_i)}$$

To avoid notational clutter, let $d(u_j, v_{j'})$ be the minimal distance between j and j'. Applying triangle inequality we get

$$\begin{aligned} d(u_j, v_{j'}) &\leq 2d_m \leq 2(2n)^{1/\alpha} \sqrt{d(u_i, v_i) \cdot d(u_j, v_j)} \\ &\leq 2(2n)^{1/\alpha} \sqrt{\frac{d(u_{j'}, v_{j'})}{n^{2/\alpha}} \cdot d(u_j, v_j)} \leq 2^{1+1/\alpha} \sqrt{d(u_j, v_j) \cdot d(u_{j'}, v_{j'})} \end{aligned}$$

Thus

$$d(u_j, v_{j'})^2 \leq \left(2^{\alpha+1}\right)^{2/\alpha} d(u_j, v_j) \cdot d(u_{j'}, v_{j'})$$

From Lemma 6 follows for $\beta < 2^{\alpha+1}$ there is at most one request $j \in R$ with $d(u_j, v_j) \geq n^{2/\alpha} \cdot d(u_i, v_i)$ that causes a relative interference of at least $1/2n$ on i under the square root power assignment. □

We now can use an almost identical approach like shown in Lemma 5 and Theorem 6 for the unidirectional case.

Lemma 8 *Let R be a request set where the length of every pair of links differs by at most factor 2 or at least $n^{2/\alpha}$. If any subset of R containing only nearly equilength requests can be scheduled with the linear power assignment using at most c colors, then all requests in R can be scheduled with $O(c)$ colors.*

Theorem 7 *Suppose there exists a ρ-approximate algorithm for the bidirectional interference scheduling problem on equilength requests. Then there exists an algorithm for the bidirectional interference scheduling problem with approximation factor $O(\rho \log n)$ for the square root power assignment.*

We omit the proofs for these lemmas, as the arguments are analogous to the unidirectional case. For scheduling the equilength request sets we again can use Algorithm 2.

Corollary 3 *The bidirectional interference scheduling problem in general metrics has an approximation factor of $O(\log^3 n)$ for the square root power assignment.*

Corollary 4 *For $\alpha > 2$, the bidirectional interference scheduling problem in the two-dimensional Euclidean space has an approximation factor of $O(\log^2 n)$ for the square root power assignment.*

2.5 The Gap of Oblivious Power Schemes

Our upper bounds on the approximation ratios of oblivious scheduling algorithms for directed communication requests depend on the aspect ratio. In this section, we show that the dependence on the aspect ratio is unavoidable. To prove this we construct a family of instances for a given oblivious power assignment function f such that using f requires at least $\Omega(n)$ colors or schedule steps while an optimum power assignment needs only $O(1)$ rounds.

Theorem 8 *Let $f : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ be any oblivious power assignment function. There exists a family of instances on a line that requires $\Omega(n)$ colors when scheduling with the powers defined by f whereas an optimal schedule has constant length.*

Proof We distinguish three cases. In the first case, we assume that f is asymptotically unbounded, that is, for every $c > 0$ and every $x_0 > 0$ there exists a value $x > x_0$ with $f(x) > c$.

We consider the following family of instances. They consist of n pairs (u_i, v_i) on a line, with distances x_i between two nodes of a pair and χy_i between neighboring pairs. Depending on β, we choose χ as a suitable constant that is large enough to get along with different values of β.

Formally, this kind of instance can be defined by $u_1, v_1, \ldots, u_n, v_n \in \mathbb{R}$ such that

$$u_i = \begin{cases} 0 & \text{if } i = 1 \\ v_{i-1} + \chi y_i & \text{otherwise} \end{cases} \qquad \text{and} \qquad v_i = u_i + x_i$$

We now define the distances x_i and y_i between the nodes recursively depending on the function f:

$$y_i = 2(x_{i-1} + y_{i-1})$$

Given $x_1, \ldots, x_{i-1}$ and y_i, we choose x_i such that $x_i \geq y_i$ and

$$f(x_i) \geq y_i^{\alpha} \frac{f(x_j)}{x_j^{\alpha}} \qquad \text{for all } j < i$$

This is always possible since f is asymptotically unbounded. By this construction it is ensured that a pair k is exposed to high interference by pairs with larger indices. To show this, let $S \subseteq [n]$ be a set of indices of pairs that can be scheduled together in one step; $k = \min S$.

For $i \in S \setminus \{k\}$ it holds that

$$d(u_i, v_k) = \sum_{j=k+1}^{i-1} x_j + \sum_{j=k+1}^{i} \chi \cdot y_j \leq 2\chi \sum_{j=k}^{i} y_j \leq 2\chi \sum_{j=k}^{i} \frac{1}{2^{i-j}} y_i \leq 4\chi y_i$$

Since all pairs in S can be scheduled in one step the SINR condition is satisfied for pair k:

$$\beta \sum_{i \in S\setminus\{k\}} \frac{p_i}{d(u_i, v_k)^\alpha} \leq \frac{p_k}{d(u_k, v_k)^\alpha} = \frac{f(x_k)}{x_k^\alpha}$$

Putting these facts together

$$\frac{1}{\beta}\frac{f(x_k)}{x_k^\alpha} \geq \sum_{i \in S\setminus\{k\}} \frac{p_i}{d(u_i, v_k)^\alpha} \geq \sum_{i \in S\setminus\{k\}} \frac{y_i^\alpha \frac{f(x_k)}{x_k^\alpha}}{(4\chi y_i)^\alpha} = \frac{|S|-1}{(4\chi)^\alpha}\frac{f(x_k)}{x_k^\alpha}$$

This implies $|S| \leq \frac{(4\chi)^\alpha}{\beta} + 1$, which means there are at least $\frac{\beta}{(4\chi)^\alpha + \beta} n = \Omega(n)$ colors needed when using $p_i = f(d(s_i, d_i))$.

On the other hand for these instances there is a power assignment, $p_i = \sqrt{2^i}$, such that there is a coloring using a constant number of colors. This is caused by the fact that for all instances described it holds that $y_i \leq x_i$ and $y_{i+1} \geq 2x_i$. Thus for any link k the interference by the ones with higher index as well as the ones with lower index forms a geometric series. This means a constant fraction of all links may have the same color and therefore there is a coloring using a constant number of colors.

In the second case, we assume that f is asymptotically bounded from above by some value $c > 0$ but does not converge to 0. In this case, there exists a value $b \in (0, c]$ such that for every $x_0 > 0$ there exists a value $x > x_0$ with $f(x) \in [b, 2b]$. Let $\chi > 1$ be a suitable constant. We choose n numbers $x_1, \ldots, x_n$ satisfying the properties (a) $f(x_i) \in [b, 2b]$, for $1 \leq i \leq n$, and (b) $x_i \geq \chi x_{i-1}$, for $2 \leq i \leq n$. We set $u_i = -x_i/2$ and $v_i = x_i/2$. This instance corresponds to nested pairs on the line, whereas the power assignment is similar to the uniform power assignment, which already indicates the desired result.

To be more precise, let $S \subseteq [n]$ be a set of indices of requests that can be scheduled together and let $k = \max S$. For $i \in S$ it holds $d(u_i, v_k) \leq x_k/2$. The SINR condition for k states

$$\beta \sum_{i\in S\setminus\{k\}} \frac{p_i}{d(u_i,v_k)^\alpha} \le \frac{p_k}{d(u_k,v_k)^\alpha} = \frac{f(x_k)}{x_k^\alpha}$$

This yields

$$\frac{1}{\beta}\cdot\frac{f(x_k)}{x_k^\alpha} \ge \sum_{i\in S\setminus\{k\}} \frac{p_i}{d(u_i,v_k)^\alpha} \ge \sum_{i\in S\setminus\{k\}} \frac{b}{(x_k/2)^\alpha} = (|S|-1)\cdot\frac{2^\alpha b}{x_k^\alpha}$$

Since $2b \ge f(x_k)$, we have $|S| \le 1/\beta \cdot 2^{1-\alpha} + 1$. It follows again that at least $\Omega(n)$ colors are needed to schedule these instances using $p_i = f(d(u_i,v_i))$.

In contrast, if χ is chosen sufficiently large than the square root power assignment can schedule all these requests simultaneously.

Finally, in the third case, $\lim f(x) = 0$, we again construct a sequence of nested pairs analogously to second case but replacing condition (a) by the condition $f(x_i) \le f(x_{i-1})$. Analogously to the second case, the power assignment defined by f allows only for scheduling a constant number of pairs simultaneously while the square root assignment can schedule all pairs simultaneously. □

The last result shows that the dependence on Δ is necessary for nontrivial results. The following theorem shows that there is a gap of at least $\Omega\left(\sqrt{\log\log\Delta}\right)$ between oblivious and optimal power assignments.

Theorem 9 *An instance of the interference scheduling problem exists such that every schedule using an oblivious power function needs at least $\Omega\left(\sqrt{\log\log\Delta}\right)$ more steps than the optimal schedule.*

Proof In this proof we construct an instance that can be scheduled in a constant number of rounds by a non-oblivious power assignment, but every oblivious power assignment needs at least $\Omega\left(\sqrt{\log\log\Delta}\right)$ steps. The instance consist of two nearly identical requests sets, only the role of sender and receiver in each request is exchanged. More formally, let $x_1 = 1$, $y_i = x_i^2$, and $x_{i+1} = 2y_i$ for every $i \in [n]$. Let the request set R_1 consist of the requests (u_i, v_i) described by

$$u_i = \begin{cases} 0 & \text{if } i = 1 \\ -\sum_{j=2}^{i} x_j & \text{otherwise} \end{cases} \qquad \text{and} \qquad v_i = \sum_{j=1}^{i} y_i$$

and let R_2 consist of requests $\left(u_i', v_i'\right)$ with

$$u_i' = M + \sum_{j=1}^{i} y_i \qquad \text{and} \qquad v_i' = \begin{cases} M & \text{if } i = 1 \\ M - \sum_{j=2}^{i} x_j & \text{otherwise} \end{cases}$$

where M denotes a constant large enough that interferences between requests from R_1 and R_2 become negligible. Since for all $i \in [n]$ holds $d(u_i,v_i) = d\left(u_i', v_i'\right)$, every oblivious power assignment uses the same power p_i for request (u_i,v_i) and $\left(u_i', v_i'\right)$.

Let T denote the schedule under an arbitrary, fixed oblivious power assignment. In this schedule there must be a step where at least n/T requests from R_1 are scheduled. Let $M \subseteq [n]$ denote their indices. Let $i, j \in M$ with $i < j$. The SINR constraint states

$$\beta \frac{p_i}{d(u_i, v_j)^\alpha} \leq \frac{p_j}{d(u_j, v_j)^\alpha}$$

Using $d(u_i, v_j) \leq x_j$ and $d(u_j, v_j) \geq y_j = x_j^2$ we get

$$\beta \frac{p_i}{x_j^\alpha} \leq \frac{p_j}{x_j^{2\alpha}}$$

which implies $p_i \leq p_j / \beta x_j^\alpha$. With $d\left(u'_j, v'_i\right) \leq 2x_j$, the interference from $\left(u'_j, v'_j\right)$ on $\left(u'_i, v'_i\right)$ is

$$\beta \frac{p_j}{d\left(u'_j, v'_i\right)^\alpha} \geq \beta \frac{p_j}{(2x_j)^\alpha} \geq \frac{\beta^2 p_i}{2^\alpha} > \frac{p_i}{d\left(u'_i, v'_i\right)}$$

Thus, for every $i \neq j, i, j \in M$, the requests $\left(u'_i, v'_i\right)$ and $\left(u'_j, v'_j\right)$ cannot be scheduled in the same step. In fact, for every $i \in M$, $\left(u'_i, v'_i\right)$ must be assigned to a different schedule step. This yields $T \geq |M|$ and it follows $T \geq \sqrt{n} = \sqrt{\Omega(\log\log \Delta)}$. □

2.6 Summary and Open Problems

We have studied the interference scheduling problem with a focus on oblivious power assignments, i.e., the power for a signal is defined as a function of the path loss. Examples of such power assignments are the uniform, the linear, and the square root power assignment. The major advantage of these power assignments is their simplicity. In particular, they can be computed for every request without taking into account other requests. In our study we investigated the approximation factors with respect to the schedule length that can be achieved with oblivious power assignments.

The linear power assignment is of special interest as it is energy efficient in the sense that signals are sent at a power level that is only a constant factor larger than the power level needed to drown out ambient noise. In Sect. 2.3, we presented lower and upper bounds for the linear power assignment from [5]. The key to both the lower and upper bounds is the measure of interference I. On the one hand, we have shown that $\Omega(I)$ is a lower bound on the schedule length when using linear power assignments. On the other hand, we have presented distributed scheduling algorithms for the linear power assignment computing schedules of length $O(I \log n)$ and $O(I + \log^2 n)$, respectively. For dense instances this gives a constant factor approximation of the optimal schedule for linear power assignment.

Similar results have been achieved recently for the uniform power assignment. In [6] it is presented an algorithm that achieves a constant factor approximation guarantee with respect to the number of requests that can be scheduled simultaneously. A straight forward extension of this approach yields an approximation factor of $O(\log n)$ with respect to the schedule length for the uniform power assignment.

How do these results compare to the schedule length for general power assignments? – In Sect. 2.3, we show a lower bound of $\Omega(I/\log \Delta \log n)$ for schedules with general power assignments, where Δ denotes the aspect ratio of the metric. When restricting to the two-dimensional Euclidean space the bound improves to $\Omega(I/\log \Delta)$. Thus, the best known scheduling algorithms for the linear and the uniform power assignments achieve approximation ratios of order $\log \Delta \text{polylog} n$ in comparison to the optimal power assignment.

In Sect. 2.4, we present an analysis showing that the square root power assignment can achieve significantly better approximation ratio in terms of the aspect ratio than the linear and the uniform power assignment: For directed communication requests the approximation ratio of the square root power assignment is of order $O(\log \Delta \text{polylog}\, n)$ and for bidirectional requests even of order only $O(\text{polylog}\, n)$. Both of these ratios compare the schedule length of the square root power assignment with the schedule length for general power assignments. The result for directed communication requests is from [9] and the result for bidirectional requests was first shown in [5] and then improved in [9].

Finally, in Sect. 2.5 we study lower bounds for oblivious power assignments. We show that the dependence on the aspect ratio cannot be avoided for directed communication requests and present a lower bound of order $\Omega\left(\sqrt{\log \log \Delta}\right)$ on the approximation ratio holding for every oblivious power assignment. In particular, one cannot achieve approximation factors better than $\Omega(n)$ for directed communication requests with unbounded aspect ratio when restricting to oblivious power assignments.

We want to conclude with the major open problems about interference scheduling in the physical model: Devise a polynomial time constant factor approximation algorithm or approximation scheme for the interference scheduling problem with general power assignments or show that such an approximation is not possible. Present improved distributed algorithms beating the currently best known approximation ratios for oblivious power assignments.

References

1. M. Andrews and M. Dinitz. Maximizing capacity in arbitrary wireless networks in the SINR model: Comple xity and game theory. In *Proceedings of the 28th Conference of the IEEE Communications Society (INFOCOM)*, Rio de Janeiro, Brazil, 2009.

2. C. Avin, Z. Lotker, and Y. A. Pignolet. On the power of uniform power: Capacity of wireless networks with bounded resources. In *Proceedings of the 17th Annual European Symposium on Algorithms (ESA)*, Copenhagen, Denmark, 2009.
3. D. Chafekar, V. S. Anil Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan. Cross-layer latency minimization in wireless networks with SINR con. In *Proceedings of the 8th ACM International Symposium Mobile Ad-Hoc Networking and Computing (MOBIHOC)*, pages 110–119, Montreal, Quebec, Canada, 2007.
4. D. Chafekar, V. S. Anil Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan. Approximation algorithms for computing capacity of wireless networks with SINR constraints. In *Proceedings of the 27th Conference of the IEEE Communications Society (INFOCOM)*, pages 1166–1174, Phoenix, AZ, USA, 2008.
5. A. Fanghänel, T. Kesselheim, H. Räcke, and B. Vöcking. Oblivious interference scheduling. In *Proceedings of the 28th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, Calgary, Alberta, Canada, 2009.
6. O. Goussevskaia, M. M. Halldórsson, R. Wattenhofer, and E. Welzl. Capacity of arbitrary wireless networks. In *Proceedings of the 28th Conference of the IEEE Communications Society (INFOCOM)*, Rio de Janeiro, Brazil, 2009.
7. O. Goussevskaia, Y. A. Oswald, and R. Wattenhofer. Complexity in geometric SINR. In *Proceedings of the 8th ACM International Symposium Mobile Ad-Hoc Networking and Computing (MOBIHOC)*, pages 100–109, Montreal, Quebec, Canada, 2007.
8. P. Gupta and P. R. Kumar. Critical power for asymptotic connectivity in wireless networks. In W. M. McEneaney, G. Yin, and Q. Zhang, editors, *Stochastic Analysis, Control, Optimization, and Applications: A Volume in Honor of W. H. Fleming*, Systems & Control: Foundations & Applications, pages 547–566. Birkhäuser, 1998.
9. M. M. Halldórsson. Wireless scheduling with power control. In *Proceedings of the 17th Annual European Symposium on Algorithms (ESA)*, Copenhagen, Denmark, 2009.
10. M. M. Halldórsson and R. Wattenhofer. Computing Wireless Capacity. In press.
11. V. S. Anil Kumar, M. V. Marathe, S. Thite, H. Balakrishnan, C. L. Barrett. The distance-2 matching problem and its relationship to the MAC-layer capacity of ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications*, 22(6):1069–1079, 2004.
12. R. Hekmat and P. Van Mieghem. Interference in wireless multi-hop ad-hoc networks and its effect on network capacity. *Wireless Networks*, 10(4):389–399, 2004.
13. S. O. Krumke, M. V. Marathe, and S. S. Ravi. Models and approximation algorithms for channel assignment in radio networks. *Wireless Networks*, 7(6):575–584, 2001.
14. T. Moscibroda and R. Wattenhofer. The complexity of connectivity in wireless networks. In *Proceedings of the 25th Conference of the IEEE Communications Society (INFOCOM)*, pages 1–13, Barcelona, Catalunya, Spain, 2006.
15. T. Moscibroda, R. Wattenhofer, and A. Zollinger. Topology control meets SINR: The scheduling complexity of arbitrary topologies. In *Proceedings of the 7th ACM International Symposium Mobile Ad-Hoc Networking and Computing (MOBIHOC)*, pages 310–321, Florence, Italy, 2006.
16. P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37(2):130–143, 1988.
17. S. Ramanathan and E. L. Lloyd. Scheduling algorithms for multi-hop radio networks. *ACM SIGCOMM Computer Communication Review*, 22(4):211–222, 1992.
18. S. Singh and C. S. Raghavendra. PAMAS—Power aware multi-access protocol with signalling for ad hoc networks. *ACM SIGCOMM Computer Communication Review*, 28(3):5–26, 1998.

Chapter 3
Maintaining Connectivity in Sensor Networks Using Directional Antennae

Evangelos Kranakis, Danny Krizanc, and Oscar Morales

Abstract Connectivity in wireless sensor networks may be established using either omnidirectional or directional antennae. The former radiate power uniformly in all directions while the latter emit greater power in a specified direction thus achieving increased transmission range and encountering reduced interference from unwanted sources. Regardless of the type of antenna being used the transmission cost of each antenna is proportional to the coverage area of the antenna. It is of interest to design efficient algorithms that minimize the overall transmission cost while at the same time maintaining network connectivity. Consider a set S of n points in the plane modeling sensors of an ad hoc network. Each sensor is equipped with a fixed number of directional antennae modeled as a circular sector with a given spread (or angle) and range (or radius). Construct a network with the sensors as the nodes and with directed edges (u, v) connecting sensors u and v if v lies within u's sector. We survey recent algorithms and study trade-offs on the maximum angle, sum of angles, maximum range, and the number of antennae per sensor for the problem of establishing strongly connected networks of sensors.

3.1 Introduction

Connectivity in wireless sensor networks is established using either omnidirectional or directional antennae. The former transmit signals in all directions while the latter within a limited predefined angle. Directional antennae can be more efficient and transmit further in a given direction for the same amount of energy than omnidirectional ones. This is due to the fact that to a first approximation the energy transmission cost of an antenna is proportional to its coverage area. To be more specific, the coverage area of an omnidirectional antenna with range r is generally modeled by a circle of radius r and consumes energy proportional to $\pi \cdot r^2$. Bycontrast, a

E. Kranakis (✉)
School of Computer Science, Carleton University, Ottawa, ON, K1S 5B6, Canada
e-mail: kranakis@scs.carleton.ca

directional antennae with angular spread φ and range R is modeled as a circular sector of angle φ and radius R and consumes energy proportional to $\varphi \cdot R^2/2$. Thus for a given energy cost E, an omnidirectional antenna can reach distance $\sqrt{E/\pi}$, while a directional antenna with angular spread φ can reach distance $\sqrt{2E/\varphi}$. We think of the directional antennae as being on a "swivel" that can be oriented toward a small target area whereas the omnidirectional antennae spread their signal in all directions. Signals arriving at a sensor within the target area of multiple antennae will interfere and degrade reception. Thus for reasons of both energy efficiency and potentially reduced interference (as well as others, e.g., security), it is tempting to replace omnidirectional with directional antennae.

Replacing omnidirectional with directional antennae: Given a set of sensors positioned in the plane with omnidirectional and/or directional antennae, a directed network is formed as follows: a directed edge is placed from sensor u to sensor v if v lies within the coverage area of u (as modeled by circles or circular sectors). Note that if the radius of all omnidirectional antennae is the same then u is in the range of v if and only if v is in the range of u, i.e., the edge is bidirectional and is usually modeled be an undirected edge.

The main issue of concern when replacing omnidirectional with directional antennae is that this may alter important characteristics such as the degree, diameter, average path length of the resulting network. For example, the first network in Figure 3.2 is strongly connected with diameter two, and more than one node can potentially transmit at the same time without interference while in the omnidirectional case (Fig. 3.1) the diameter is one but only one antennae can transmit at a time without interference. In addition, and depending on the breadth and range of the directional antennae, the original topology depicted in Fig. 3.1 can be obtained only by using more than one directional antenna per sensor (see Fig. 3.3).

Replacing omnidirectional with directional antennae enables the sensors to reach farther using the same energy consumption. As an example consider the graphs depicted in Figs. 3.4 and 3.5. The line graph network in Fig. 3.4 with undirected edges $\{1,2\}$, $\{2,3\}$, $\{3,4\}$ is replaced by a network of directional antennae depicted in Fig. 3.5 and having $(1,2)$, $(1,3)$, $(2,3)$, $(2,4)$, $(3,4)$, $(4,3)$, $(4,2)$, $(3,2)$, $(3,1)$ as directed edges. By setting the angular spread of the directional antennae to be small a significant savings in energy are possible.

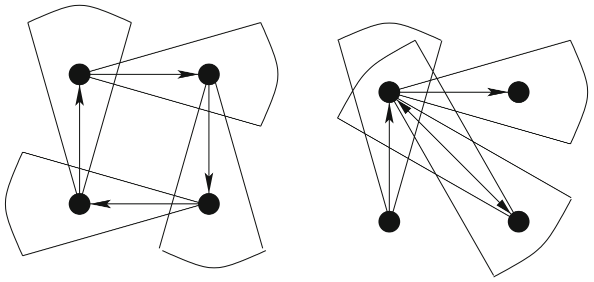


Fig. 3.1 Four sensors using directional antennae. For the same set of points, the resulting directed graphs depend on the antennae orientations

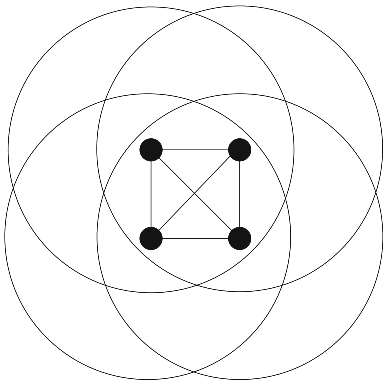


Fig. 3.2 Four sensors using omnidirectional antennae. They form an underlying complete network on four nodes

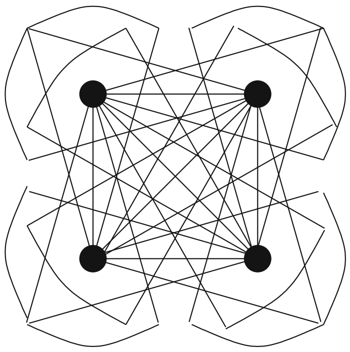


Fig. 3.3 Four sensors using directional antennae. Using three directional antennae per sensor in order to form an underlying complete network on four nodes

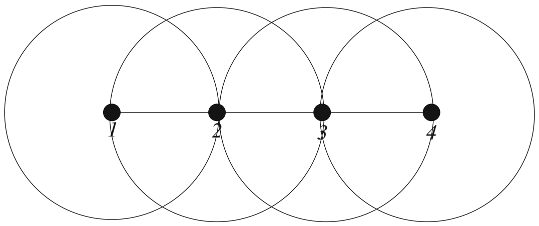


Fig. 3.4 Line graph network with undirected edges {1, 2}, {2, 3}, {3, 4} resulting when four sensors 1, 2, 3, 4 use omnidirectional antennae

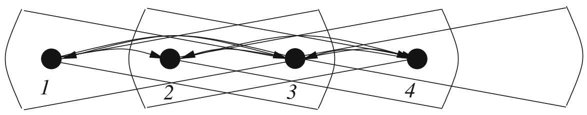


Fig. 3.5 Directed network resulting from Fig. 3.4 when the four sensors replace omnidirectional with directional antennae. Sensor number 3 is using two directional antennae while the rest only one

3.1.1 Antenna Orientation Problem

The above considerations lead to numerous questions concerning trade-offs between various factors, such as connectivity, diameter, interference, when using directional versus omnidirectional antennae in constructing sensor networks. Here we study how to maintain network connectivity when antennae angles are being reduced while at the same time the transmission range of the sensors is being kept as low as possible. More formally this raises the following optimization problem.

> Consider a set S of n points in the plane that can be identified with sensors having a range $r > 0$. For a given angle $0 \leq \varphi \leq 2\pi$ and integer k, each sensor is allowed to use at most k directional antennae each of angle at most φ. Determine the minimum range r required so that by appropriately rotating the antennae, a directed, strongly connected network on S is formed.

Note that the range of a sensor must be at least the length of the longest edge of a minimum spanning tree on the set S, since this is the smallest range required just to attain connectivity.

3.1.2 Preliminaries and Notation

Consider a set S of n points in the plane and an integer $k \geq 1$. We give the following definitions.

Definition 1 $r_k(S, \varphi)$ is the minimum range of directed antennae of angular spread at most φ so that if every sensor in S uses at most k such antennae (under an appropriate rotation) a strongly connected network on S results.

A special case is when $\varphi = 0$, for which we use the simpler notation $r_k(S)$ instead of $r_k(S, 0)$. Clearly, different directed graphs can be produced depending on the range and direction of the directional antennae. This gives rise to the following definition.

Definition 2 Let $\mathcal{D}_k(S)$ be the set of all strongly connected graphs on S with out-degree at most k.

For any graph $G \in \mathcal{D}_k(S)$, let $r_k(G)$ be the maximum length of an edge in G. It is easy to see that $r_k(S) := \min_{G \in \mathcal{D}_k(S)} r_k(G)$. It is useful to relate $r_k(S)$ to another quantity which arises from a minimum spanning tree (MST) on S.

Definition 3 Let MST (S) denote the set of all MSTs on S.

Definition 4 For $T \in \text{MST}(S)$ let $r(T)$ denote the length of longest edge of T and let $r_{\text{MST}}(S) = \min\{r(T) : T \in \text{MST}(S)\}$.

For a set S of size n, it is easily seen that $r_{\text{MST}}(S)$ can be computed in $O(n^2)$ time. Further, for any angle $\varphi \geq 0$, it is clear that $r_{\text{MST}}(S) \leq r_k(S, \varphi)$ since every strongly connected, directed graph on S has an underlying spanning tree.

3.1.3 Related Work

When each sensor has one antenna and the angle $\varphi = 0$ then our problem is easily seen to be equivalent to finding a Hamiltonian cycle that minimizes the maximum length of an edge. This is the well-known bottleneck traveling salesman problem.

3.1.3.1 Bottleneck Traveling Salesman Problem

Let $1, 2, \ldots, n$ be a set of n labeled vertices with associated edge weights $w(i, j)$, for all i, j. The *bottleneck traveling salesman problem (BTSP)* asks to find a Hamiltonian cycle in the complete (weighted) graph on the n points which minimizes the maximum weight of an edge, i.e.,

$$\min\{\max_{(i,j)\in H} w(i, j) : H \text{ is a Hamiltonian cycle}\}$$

Parker and Rardin [31] study the case where the weights satisfy the triangle inequality and they give a two-approximation algorithm for this problem. (They also show that no polynomial time $(2 - \varepsilon)$-approximation algorithm is possible for metric BTSP unless $P = NP$.) Clearly, their approximation result applies to our problem for the special case of one antennae and $\varphi = 0$. The proof uses a result in [12] that the square of every two-connected graph is Hamiltonian. (The square $G^{(2)}$ of a graph $G = (V, E)$ has the same node set V and edge set $E^{(2)}$ defined by $\{u, v\} \in E^{(2)} \Leftrightarrow \exists w \in V(\{u, w\} \in E \,\&\, \{w, v\} \in E)$.) In fact the latter paper also gives an algorithm for constructing such a Hamiltonian cycle. A generalization of this problem to finding strongly connected subgraphs with minimum maximum edge weight is studied by Punnen [32].

3.1.3.2 MST and Out-Degrees of Nodes

It is easy to see that the degree structure of an MST on a point set is constrained by proximity. If a vertex has many neighbors then some of them have to be too close together and can thus be connected directly. This can be used to show that for a given point set there is always a Euclidean minimum spanning tree of maximum degree six. In turn, this can be improved further to provide an MST with max degree five [28]. Since for large enough r every set of sensors in the plane has a Euclidean spanning tree of degree at most five and maximum range r, it is easy to see that given such minimum r and $k \geq 5$, $r_k(S) = r$. A useful parameter is the maximum degree of a spanning tree. This gives rise to the following definition.

Definition 5 For $k \geq 2$, a maximum degree k spanning tree (abbreviated $Dk - ST$) is a spanning tree all of whose vertices have degree at most k.

Related literature concerns trade-offs between maximum degree and minimum weight of the spanning tree. For example, [2] gives a quasi-polynomial time approximation scheme for the minimum weight Euclidean $D3 - ST$. Similarly, [21] and

[6] obtain approximations for minimum weight $D3-ST$ and $D4-ST$. In addition, [13] shows that it is an NP-hard problem to decide for a given set S of n points in the Euclidean plane and a given real parameter w, whether S admits a spanning tree of maximum node degree four (i.e., $D4-ST$) whose sum of edge lengths does not exceed w. Related is also [22] which gives a simple algorithm to find a spanning tree that simultaneously approximates a shortest path tree and a minimum spanning tree. In particular, given the two trees and a $\gamma > 0$, the algorithm returns a spanning tree in which the distance between any node and the root is at most $1+\gamma\sqrt{2}$ times the shortest path distance, and the total weight of the tree is at most $1+\sqrt{2}/\gamma$ times the weight of a minimum spanning tree.

Of interest here is the connection between strongly connected geometric spanners with given out-degree on a point set and the maximum length edge of an MST. Beyond the connection of BTSP mentioned above we know of no other related literature on this specific question.

3.1.3.3 Enhancing Network Performance Using Directional Antennae

Directional antennae are known to enhance ad hoc network capacity and performance and when replacing omnidirectional with directional antennae one can reduce the total energy consumption of the network. A theoretical model to this effect is presented in [16] showing that when n omnidirectional antennae are optimally placed and assigned optimally chosen traffic patterns the transport capacity is $\Theta\left(\sqrt{W/n}\right)$, where each antenna can transmit W bits per second over the common channel(s). When both transmission and reception are directional, [39] proves an $\sqrt{2\pi/\alpha\beta}$ capacity gain as well as corresponding throughput improvement factors, where α is the transmission angle and β is a parameter indicating that $\beta/2\pi$ is the average proportion of the number of receivers inside the transmission zone that will get interfered with.

Additional experimental studies confirm the importance of using directional antennae in ad hoc networking for enhancing channel capacity and improving multiaccess control. For example, research in [33] considers several enhancements, including "aggressive" and "conservative" channel access models for directional antennae, link power control, and neighbor discovery and analyzes them via simulation. [38] and [37] consider how independent communications between directional antennae can occur in parallel and calculate interference-based capacity bounds for a generic antenna model as well as a real-world antenna model and analyze how these bounds are affected by important antenna parameters like gain and angle. The authors of [3] propose a distributed receiver-oriented multiple access (ROMA) channel access scheduling protocol for ad hoc networks with directional antennae, each of which can form multiple beams and commence several simultaneous communication sessions. Finally, [24] considers energy consumption thresholds in conjunction to k-connectivity in networks of sensors with omnidirectional and directional antennae, while [23] studies how directional antennae affect overall coverage and connectivity.

A related problem that has been addressed in the literature is one that studies connectivity requirements on undirected graphs that will guarantee highest edge connectivity of its orientation, c.f. [29] and [14].

3.1.3.4 Other Applications

It is interesting to note that beyond reducing the energy consumption, directional antennae can enhance security. Unlike omnidirectional antennae that spread their signal in all directions over an angle 2π, directional antennae can attain better security because they direct their beam toward the target thus avoiding potential risks along the transmission path. In particular, in a hostile environment a directional antenna can decrease the radiation region within which nodes could receive the electromagnetic signals with high quality. For example, this has led [17] to the design of several authentication protocols based on directional antennae. In [27] they employ the average probability of detection to estimate the overall security benefit level of directional transmission over the omnidirectional one. In [18] they examine the possibility of key agreement using variable directional antennae. In [30] the use of directional antennae and beam steering techniques in order to improve performance of 802.11 links is investigated in the context of communication between a moving vehicle and roadside access points.

3.1.4 Outline of the Presentation

The following is an outline of the main issues that will be addressed in this survey. In Sect. 3.2 we discuss approximation algorithms to the main problem introduced above. The constructions are mainly based on an appropriately defined MST of the set of points. Section 3.2.1 focuses on the case of a single antenna per sensor while Sect. 3.2.2 on k antennae per sensor, for a given $2 \leq k \leq 4$. (Note that the case $k \geq 5$ is handled by using a degree five MST.) In Sect. 3.3 we discuss NP-completeness results for the cases of one and two antennae. In Sect. 3.4 we investigate a variant of the main problem whereby we want to minimize the sum of the angles of the antennae given a bound on their radius. Unlike Sect. 3.2 where we have the flexibility to select and adapt an MST on the given point set S, Sect. 3.5 considers the case whereby the underlying network is given in advance as a planar spanner on the set S and we study number of antennae and stretch factor trade-offs between the original graph and the resulting planar spanner. In addition throughout the chapter we propose several open problems and discuss related questions of interest.

3.2 Orienting the Sensors of a Point Set

In this section we consider several algorithms for orienting antennae so that the resulting spanner is strongly connected. Moreover we look at trade-offs between antenna range and breadth.

3.2.1 Sensors with One Antenna

The first paper to address the problem of converting a connected (undirected) graph resulting from omnidirectional sensors to a strongly connected graph of directional sensors having only one directional antenna each was [5].

3.2.1.1 Sensors on the Line

The first scenario to be considered is for sensors on a line. Assume that each sensor's directional antenna has angle φ. Further assume that $\varphi \geq \pi$. The problem of minimizing the range in this case can be seen to be equivalent to the same problem for the omnidirectional case, simply by pointing the antennae so as to cover the same nodes as those covered by the omnidirectional antenna as depicted in Fig. 3.6. Clearly a range equal to the maximum distance between any pair of adjacent sensors is necessary and sufficient.

When the angle φ of the antennae is less than π then a slightly more complicated orientation of the antennae is required so as to achieve strong connectivity with minimum range.

Theorem 1 ([5]) *Consider a set of $n > 2$ points x_i, $i = 1, 2, \ldots, n$, sorted according to their location on the line. For any $\pi > \varphi \geq 0$ and $r > 0$, there exists an orientation of sectors of angle φ and radius r at the points so that the transmission graph is strongly connected if and only if the distance between points i and $i + 2$ is at most r, for any $i = 1, 2, \ldots, n - 2$.*

Proof Assume $d(x_i, x_{i+2}) > r$, for some $i \leq n - 2$. Consider the antenna at x_{i+1}. There are two cases to consider. First, if the antenna at x_{i+1} is directed to the left then the portion of the graph to its left cannot be connected to the portion of the graph to the right; second, if the antenna at x_{i+1} is directed to the right then the portion of the graph to its right cannot be connected to the portion of the graph to the left. In either case the graph becomes disconnected.

Conversely, assume $d(x_i, x_{i+2}) \leq r$, for all $i \leq n - 2$. Consider the following antenna orientation for an even number of sensors (see Fig. 3.7). (The odd case is handled similarly.)

1. antennas $x_1, x_3, x_5, \ldots$ labeled with odd integers are oriented right and
2. antennas $x_2, x_4, x_6, \ldots$ labeled with even integers are oriented left.

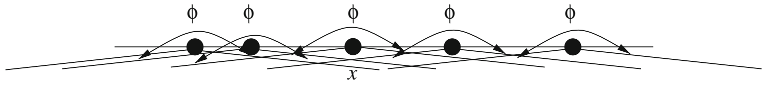


Fig. 3.6 Antenna orientation for a set of sensors on a line when the angle $\boldsymbol{\varphi} \geq \boldsymbol{\pi}$

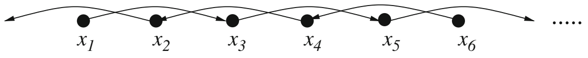


Fig. 3.7 Antenna orientation for a set of sensors on a line when the angle $\boldsymbol{\varphi} < \boldsymbol{\pi}$

It is easy to see that the resulting orientation leads to a strongly connected graph. This completes the proof of Theorem 1.

3.2.1.2 Sensors on the Plane

The case of sensors on the plane is more challenging. As was noted above the case of $\varphi = 0$ is equivalent to the Euclidean BTSP and thus the minimum range can be approximated to within a factor of 2. In [5] the authors present a polynomial time algorithm for the case when the sector angle of the antennae is at least $8\pi/5$. For smaller sector angles, they present algorithms that approximate the minimum radius. We present the proof of this last result below.

Theorem 2 (Caragiannis et al. [5]) *Given an angle φ with $\pi \leq \varphi < 8\pi/5$ and a set S of points in the plane, there exists a polynomial time algorithm that computes an orientation of sectors of angle φ and radius $2\sin\left(\pi - \frac{\varphi}{2}\right) \cdot r_1(S, \varphi)$ so that the transmission graph is strongly connected.*

Proof Consider a set S of nodes on the Euclidean plane and let T be a minimum spanning tree of S. Let $r = r_{\text{MST}}(S)$ be the longest edge of T. We will use sectors of angle φ and radius $d(\varphi) = 2r \sin\left(\pi - \frac{\varphi}{2}\right)$ and we will show how to orient them so that the transmission graph induced is a strongly connected subgraph over S. The theorem will then follow since r is a lower bound on $r_1(S, \varphi)$.

We first construct a matching M consisting of (mutually non-adjacent) edges of T with the following additional property: any non-leaf node of T is adjacent to an edge of M. This can be done as follows. Initially, M is empty. We root T at an arbitrary node s. We pick an edge between s and one of its children and insert it in M. Then, we visit the remaining nodes of T in a BFS (breadth first search) manner. When visiting a node u, if u is either a leaf node or a non-leaf node such that the edge between it and its parent is in M, we do nothing. Otherwise, we pick an edge between u and one of its children and insert it to M.

We denote by Λ the leaves of T which are not adjacent to edges of M. We also say that the endpoints of an edge in M form a *couple*. We use sectors of angle φ and radius $d(\varphi)$ at each point and orient them as follows. At each node $u \in \Lambda$, the sector is oriented so that it induces the directed edge from u to its parent in T in the corresponding transmission graph G. For each two points u and v forming a couple, we orient the sector at u so that it contains all points p at distance $d(\varphi)$ from u for which the counterclockwise angle $v\hat{u}p$ is in $[0, \varphi]$, see Fig. 3.8.

We first show that the transmission graph G defined in this way has the following property, denoted by **(P)**, and stated in the Claim below.

*Claim (***P***)* . For any two points u and v forming a couple, G contains the two opposite directed edges between u and v, and, for each neighbor w of either u or v in T, it contains a directed edge from either u or v to w.

Consider a point w corresponding to a neighbor of u in T (the argument for the case where w is a neighbor of v is symmetric). Clearly, w is at distance $|uw| \leq r$ from u. Also, note that since $\varphi < 8\pi/5$, we have that the radius of the sectors is

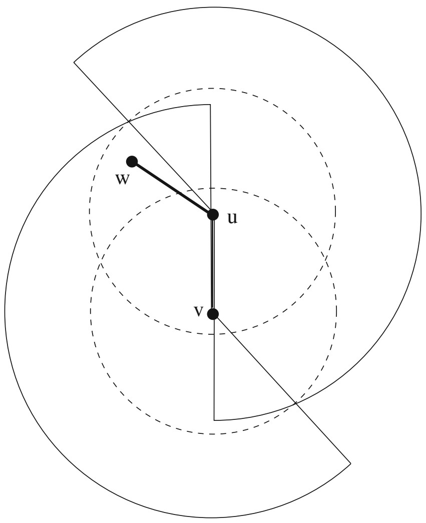


Fig. 3.8 The orientation of sectors at two nodes u, v forming a couple, and a neighbor w of u that is not contained in the sector of u. The *dashed circles* have radius r and denote the range in which the neighbors of u and v lie

$d(\varphi) = 2r \sin\left(\pi - \frac{\varphi}{2}\right) \geq 2r \sin \frac{\pi}{5} > 2r \sin \frac{\pi}{6} = r$. Hence, w is contained in the sector of u if the counterclockwise angle $v\hat{u}w$ is at most φ; in this case, the graph G contains a directed edge from u to w. Now, assume that the angle $v\hat{u}w$ is $x > \varphi$ (see Fig. 3.8). By the law of cosines in the triangle defined by points u, v, and w, we have that

$$\begin{aligned}
|vw| &= \sqrt{|uw|^2 + |uv|^2 - 2|uw||uv| \cos x} \\
&\leq r\sqrt{2 - 2\cos x} \\
&= 2r \sin \frac{x}{2} \\
&\leq 2r \sin\left(\pi - \frac{\varphi}{2}\right) \\
&= d(\varphi)
\end{aligned}$$

Since the counterclockwise angle $v\hat{u}w$ is at least π, the counterclockwise angle $u\hat{v}w$ is at most $\pi \leq \varphi$ and, hence, w is contained in the sector of v; in this case, the graph G contains a directed edge from v to w. In order to complete the proof of property (**P**), observe that since $|uv| \leq r \leq d(\varphi)$ the point v is contained in the sector of u (and vice versa).

Now, in order to complete the proof of the theorem, we will show that for any two neighbors u and v in T, there exist a directed path from u to v and a directed path from v to u in G. Without loss of generality, assume that u is closer to the root s of T than v. If the edge between u and v belongs to M (i.e., u and v form a couple), property (**P**) guarantees that there exist two opposite directed edges between u and v in the transmission graph G. Otherwise, let w_1 be the node with which u forms a couple. Since v is a neighbor of u in T, there is either a directed edge from u to v

in G or a directed edge from w_1 to v in G. Then, there is also a directed edge from u to w_1 in G which means that there exists a directed path from u to v. If v is a leaf (i.e., it belongs to Λ), then its sector is oriented so that it induces a directed edge to its parent u. Otherwise, let w_2 be the node with which v forms a couple. Since u is a neighbor of v in T, there is either a directed edge from v to u in G or a directed edge from w_2 to u in G. Then, there is also a directed edge from v to w_2 in G which means that there exists a directed path from v to u.

3.2.1.3 Further Questions and Open Problems

In Sect. 3.3 we present a lower bound from [5] that shows this problem is NP-hard for angles smaller than $2\pi/3$. This leaves the complexity of the problem open for angles between $2\pi/3$ and $8\pi/5$. Related problems that deserve investigation include the complexity of gossiping and broadcasting as well as other related communication tasks in this geometric setting.

3.2.2 Sensors with Multiple Antennae

We are interested in the problem of providing an algorithm for orienting the antennae and ultimately for estimating the value of $r_k(S, \varphi)$. Without loss of generality antennae ranges will be normalized to the length of the longest edge in any MST, i.e., $r_{\mathrm{MST}}(S) = 1$. The main result concerns the case $\varphi = 0$ and was proven in [10]:

Theorem 3 (Dobrev et al. [10]) *Consider a set S of n sensors in the plane, and suppose each sensor has k, $1 \le k \le 5$, directional antennae. Then the antennae can be oriented at each sensor so that the resulting spanning graph is strongly connected and the range of each antenna is at most $2 \cdot \sin\left(\frac{\pi}{k+1}\right)$ times the optimal. Moreover, given an MST on the set of points the spanner can be constructed with additional $O(n)$ overhead.*

The proof in [10] considers five cases depending on the number of antennae that can be used by each sensor. As noted in the introduction, the case $k = 1$ was derived in [31]. The case $k = 5$ follows easily from the fact that there is an MST with maximum vertex degree five. This leaves the remaining three cases for $k = 2, 3, 4$. Due to space limitations we will not give the complete proof here. Instead we will discuss only the simplest case $k = 4$.

3.2.2.1 Preliminary Definitions

Before proceeding with presentation of the main results we introduce some notation which is specific to the following proofs. $D(u; r)$ is the open disk with radius r. $d(\cdot, \cdot)$ denotes the usual Euclidean distance between two points. In addition, we define the concept of *Antenna-Tree* (*A-Tree*, for short) which isolates the particular properties of an MST that we need in the course of the proof.

Definition 6 An A-Tree is a tree T embedded in the plane satisfying the following three rules:

1. Its maximum degree is five.
2. The minimum angle among nodes with a common parent is at least $\pi/3$.
3. For any point u and any edge $\{u, v\}$ of T, the open disk $D(v; d(u, v))$ does not have a point $w \neq v$ which is also a neighbor of u in T.

It is well known and easy to prove that for any set of points there is an MST on the set of points which satisfies Definition 6. Recall that we consider normalized ranges (i.e., we assume $r(T) = 1$).

Definition 7 For each real $r > 0$, we define the *geometric rth power* of a A-Tree T, denoted by $T^{(r)}$, as the graph obtained from T by adding all edges between vertices of (Euclidean) distance at most r.

For simplicity, in the sequel we slightly abuse terminology and refer to the geometric rth power as the *rth power*.

Definition 8 Let G be a graph. An orientation $\overrightarrow{G}$ of G is a digraph obtained from G by orienting every edge of G in at least one direction.

As usual, we denote with (u, v) a directed edge from u to v, whereas $\{u, v\}$ denotes an undirected edge between u and v. Let $d^+(\overrightarrow{G}, u)$ be the out-degree of u in $\overrightarrow{G}$ and $\Delta^+(\overrightarrow{G})$ the maximum out-degree of a vertex in $\overrightarrow{G}$.

3.2.2.2 Maximum Out-Degree four

In this section we prove that there always exists a subgraph of $T^{(2\sin\pi/5)}$ that can be oriented in such a way that it is strongly connected and its maximum out-degree is four. A precise statement of the theorem is as follows.

Theorem 4 (Dobrev et al. [10]) *Let T be an A-Tree. Then there exists a spanning subgraph $G \subseteq T^{(2\sin\pi/5)}$ such that $\overrightarrow{G}$ is strongly connected and $\Delta^+(\overrightarrow{G}) \leq 4$. Moreover, $d^+(\overrightarrow{G}, u) \leq 1$ for each leaf u of T and every edge of T incident to a leaf is contained in G.*

Proof We first introduce a definition that we will use in the course of the proof. We say that two consecutive neighbors of a vertex are *close* if the smaller angle they form with their common vertex is at most $2\pi/5$. Observe that if v and w are close, then $|v, w| \leq 2\sin\pi/5$. In all the figures in this section an angular sign with a dot depicts close neighbors. The proof is by induction on the diameter of the tree. First, we do the base case. Let k be the diameter of T. If $k \leq 1$, let $G = T$ and the result follows trivially. If $k = 2$, then T is an A-Tree which is a star with $2 \leq d \leq 5$ leaves. Two cases can occur:

- $d < 5$. Let $G = T$ and orient every edge in both directions. This results in a strongly connected digraph which trivially satisfies the hypothesis of the theorem.

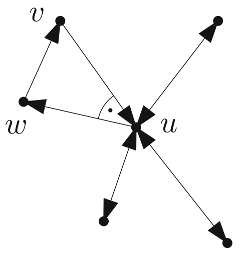


Fig. 3.9 T is a tree with five leaves and diameter $k = 2$

- $d = 5$. Let u be the center of T. Since T is a star, two consecutive neighbors of u, say, v and w, are close. Let $G = T \cup \{\{v, w\}\}$ and orient edges of G as depicted in Fig. 3.9.[1] It is easy to check that G satisfies the hypothesis of the theorem.

Next we continue with the inductive step. Let T' be the tree obtained from T by removing all leaves. Since removal of leaves does not violate the property of being an A-Tree, T' is also an A-Tree and has diameter less than the diameter of T. Thus, by inductive hypothesis there exists $G' \subseteq T'^{(2\sin\pi/5)}$ such that $\overrightarrow{G'}$ is strongly connected, $\Delta^+(\overrightarrow{G'}) \leq 4$. Moreover, $d^+(\overrightarrow{G}, u) \leq 1$ for each leaf u of T' and every edge of T' incident to a leaf is contained in G'.

Let u be a leaf of T', u_0 be the neighbor of u in T', and $u_1, .., u_c$ be the c neighbors of u in $T \setminus T'$ in clockwise order around u starting from u_0. Two cases can occur:

- $c \leq 3$. Let $G = G' \cup \{\{u, u_1\}, .., \{u, u_c\}\}$ and orient these c edges in both directions. $\overrightarrow{G}$ satisfies the hypothesis since $G \subseteq T^{(2\sin\pi/5)}$, $\Delta^+(\overrightarrow{G}) \leq 4$, $d^+(\overrightarrow{G}, u) \leq 1$ for each leaf u of T and every edge of T incident to a leaf is contained in G.
- $c = 4$. We consider two cases. In the first case suppose that two consecutive neighbors of u in $T \setminus T'$ are close. Consider u_k and u_{k+1} are close, where $1 \leq k < 4$. Define $G = G' \cup \{\{u, u_1\}, \{u, u_2\}, \{u, u_3\}, \{u, u_4\}, \{u_k, u_{k+1}\}\}$ and orient edges of G as depicted in Fig. 3.10. In the second case, either u_0 and u_1 are close or u_0 and u_4 are close. Without loss of generality, let assume that u_0 and u_1 are close. Thus, let $G = \{G' \setminus \{u, u_0\}\} \cup \{\{u, u_1\}, \{u, u_2\}, \{u, u_3\}, \{u, u_4\}, \{u_0, u_1\}\}$, but now the orientation of G will depend on the orientation of $\{u, u_0\}$ in G'. Thus,

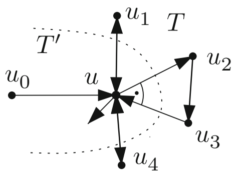


Fig. 3.10 Depicting the inductive step when u has four neighbors in $T' \setminus T$ and u_k and u_{k+1} are close, where $k = 2$. (The *dotted curve* is used to separate the tree T' from T)

[1] In all figures boldface arrows represent the newly added edges.

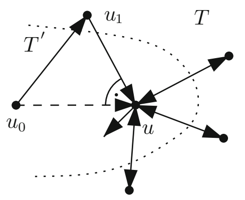


Fig. 3.11 Depicting the inductive step when u has four neighbors in $T' \setminus T$, u_0 and u_1 are close and (u_0, u) is in the orientation of G' (The *dashed edge* $\{u_0, u\}$ indicates that it does not exist in G but exists in G' and the *dotted curve* is used to separate the tree T' from T)

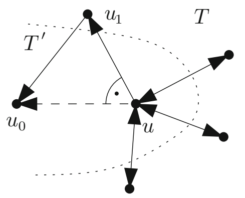


Fig. 3.12 Depicting the inductive step when u has four neighbors in $T' \setminus T$, u_0 and u_1 are close and (u, u_0) is in the orientation of G' (The *dashed edge* $\{u_0, u\}$ indicates that it does not exist in G but exists in G' and the *dotted curve* is used to separate the tree T' from T)

if (u_0, u) is in $\overrightarrow{G'}$, then orient edges of G as depicted in Fig. 3.11. Otherwise if (u, u_0) is in $\overrightarrow{G'}$, then orient edges of G as depicted in Fig. 3.12. $\overrightarrow{G}$ satisfies the hypothesis since $G \subseteq T^{(2 \sin \pi/5)}$, $\Delta^+(\overrightarrow{G}) \leq 4$, $d^+(\overrightarrow{G}, u) \leq 1$ for each leaf u of T and every edge of T incident to a leaf is contained in G.

This completes the proof of the theorem.

The above implies immediately the case $k = 4$ of Theorem 3. The remaining cases of $k = 3$ and $k = 2$ are similar but more complex. The interested reader can find details in [10].

3.2.2.3 Further Questions and Open Problems

There are several interesting open problems all related to the optimality of the range $2 \sin \left(\frac{\pi}{k+1} \right)$ which was derived in Theorem 3. This value is obviously optimal for $k = 5$ but the cases $1 \leq k \leq 4$ remain open. Additional questions concern studying the problem in d-dimensional Euclidean space, $d \geq 3$, and more generally in normed spaces. The case $d = 3$ would also be of particular interest to sensor networks.

3.3 Lower Bounds

In this section we discuss the only known lower bounds for the problem.

3.3.1 One Antenna Per Sensor

When the sector angle is smaller than $2\pi/3$, the authors of [5] show that the problem of determining the minimum radius in order to achieve strong connectivity is NP-hard.

Theorem 5 (Caragiannis et al. [5]) *For any constant $\varepsilon > 0$, given φ such that $0 \leq \varphi < 2\pi/3 - \varepsilon$, $r > 0$, and a set of points on the plane, determining whether there exists an orientation of sectors of angle φ and radius r so that the transmission graph is strongly connected is NP-complete.*

A simple proof is by reduction from the well-known NP-hard problem for finding Hamiltonian cycles in degree three planar graphs [15]. In particular, a weaker statement for sector angles smaller than $\pi/2$ follows by the same reduction used in [19] in order to prove that the Hamiltonian circuit problem in grid graphs is NP-complete. Consider an instance of the problem consisting of points with integer coordinates on the Euclidean plane (these can be thought of as the nodes of the grid proximity graph between them). Then, if there exists an orientation of sector angles of radius 1 and angle $\varphi < \pi/2$ at the nodes so that the corresponding transmission graph is strongly connected, then this must also be a Hamiltonian circuit of the proximity graph. The construction of [19] can be thought of as reducing the Hamiltonian circuit problem on bipartite planar graphs of maximum degree three (which is proved in [19] to be NP-complete) to an instance of the problem with a grid graph as a proximity graph such that there exists a Hamiltonian circuit in the grid graph if and only if the original graph has a Hamiltonian circuit. The proof of [5] uses a slightly more involved reduction with different gadgets in order to show that the problem is NP-complete for sector angles smaller than $2\pi/3$.

3.3.2 Two Antennae Per Sensor

For two antennae the best known lower bound is from [10] and can be stated as follows.

Theorem 6 (Dobrev et al. [10]) *For $k = 2$ antennae, if the angular sum of the antennae is less than α then it is NP-hard to approximate the optimal radius to within a factor of x, where x and α are the solutions of equations $x = 2\sin(\alpha) = 1 + 2\cos(2\alpha)$.*

Observe that by using the identity $\cos(2\alpha) = 1 - 2\sin^2\alpha$ above and by solving the resulting quadratic equation with unknown $\sin\alpha$ we obtain numerical solutions $x \approx 1.30, \alpha \approx 0.45\pi$.

As before, the proof is by reduction from the well-known NP-hard problem for finding Hamiltonian cycles in degree three planar graphs [15]. In particular, the construction in [10] takes a degree three planar graph $G = (V, E)$ and replaces each vertex $v \in V$ by a vertex graph (meta-vertex) G_v and each edge $e \in E$ of G by an

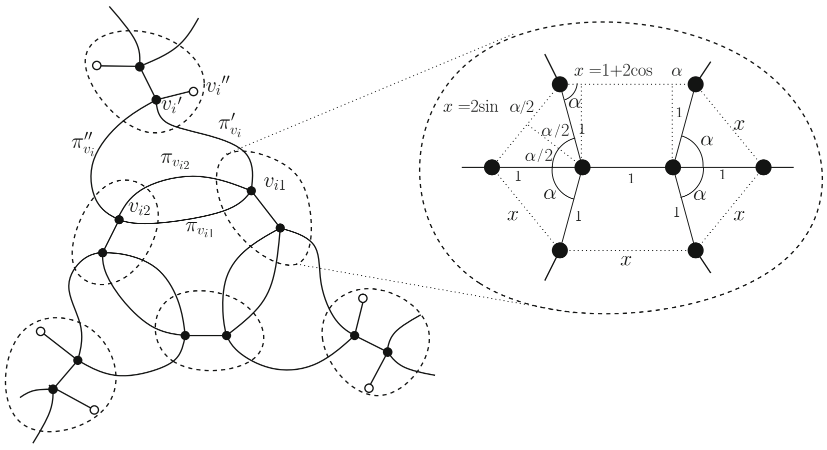


Fig. 3.13 Connecting meta-edges with meta-vertices. The *dashed ovals* show the places where embedding is constrained

edge graph (meta-edge) G_e. Figure 3.13 shows that how meta-edges are connected with meta-vertices. Further details of the construction can be found in [10].

3.3.2.1 Further Questions and Open Problems

It is interesting to note that in addition to the question of improving the lower bounds in Theorems 5 and 6 no lower bound or NP-completeness result is known for the cases of three or four antennae.

3.4 Sum of Angles of Antennae

A variant of the main problem is considered in a subsequent paper [4]. As before each sensor has fixed number of directional antennae and we are interested in achieving strong connectivity while minimizing the sum (taken over all sensors) of angles of the antennae under the assumption that the range is set at the length of the longest edge in any MST (normalized to 1). The authors present trade-offs between the antennae range and specified sums of antennae, given that we have k directional antennae per sensor for $1 \le k \le 5$. The following result is proven in [4].

Lemma 1 (Caragiannis et al. [4]) *Assume that a node u has degree d and the sensor at u is equipped with k antennae, where $1 \le k \le d$, of range at least the maximum edge length of an edge from u to its neighbors. Then $2(d-k)\pi/d$ is always sufficient and sometimes necessary bound on the sum of the angles of the antennae at u so that there is an edge from u to all its neighbors in an MST.*

Proof The result is trivially true for $k = d$ since we can satisfy the claim by directing a separate antenna of angle 0 to each node adjacent to u. So we can assume that

$k \leq d - 1$. To prove the necessity of the claim take a point at the center of a circle and with d adjacent neighbors forming a regular d-gon on the perimeter of the circle of radius equal to the maximum edge length of the given spanning tree on S. Thus each angle formed between two consecutive neighbors on the circle is exactly $2\pi/d$. It is easy to see that for this configuration a sum of $2(d-k)\pi/d$ is always necessary.

To prove that sum $2(d-k)\pi/d$ is always sufficient we argue as follows. Consider the point u which has d neighbors and consider the sum $\boldsymbol{\sigma}$ of the largest k angles formed by $k+1$ consecutive points of the regular polygon on the perimeter of the circle. We claim that $\sigma \geq 2k\pi/d$. Indeed, let the d consecutive angles be $\alpha_0, \alpha_1, ..., \alpha_{d-1}$. (see Fig. 3.14). Consider the d sums $\alpha_i + \alpha_{i+1} + ... + \alpha_{i+k-1}$, for $i = 0, ..., d-1$, where addition on the indices is modulo d. Observe, that

$$2k\pi = \sum_{i=0}^{d-1}(\alpha_i + \alpha_{i+1} + ... + \alpha_{i+k-1}) \leq d\sigma$$

It follows that the remaining angles sum to at most $2\pi - \sigma \leq 2\pi - 2k\pi/d = 2\pi(d-k)/d$. Now consider the $k+1$ consecutive points, say $p_1, p_2, ..., p_{k+1}$, such that the sum σ of the k consecutive angles formed is at least $2k\pi/d$. Use $k-1$ antennae each of size 0 rad to cover each of the points $p_2, ..., p_k$, respectively, and an angle of size $2\pi(d-k)/d$ to cover the remaining $n-k+1$ points. This proves the lemma.

The next simple result is an immediate consequence of Lemma 1 and indicates how antennae spreads affect the range in order to accomplish strong connectivity.

Definition 9 Let φ_k be a given non-negative value in $[0, 2\pi)$ such that the sum of angles of k antennae at each sensor location is bounded by φ_k. Further, let r_{k,φ_k} denote the minimum radius (or range) of directional antennae for a given k and φ_k that achieves strong connectivity under some rotation of the antennae.

We can prove the following result.

Theorem 7 (Caragiannis et al. [4]) *For any* $1 \leq k \leq 5$*, if* $\varphi_k \geq \frac{2(5-k)\pi}{5}$ *then* $r_{k,\varphi_k} = 1$.

Proof We prove the theorem by showing that if $\varphi_k \geq 2(5-k)\pi/5$ then the antennae can be oriented in such a way that for every vertex u there is a directed edge from u to all its neighbors.

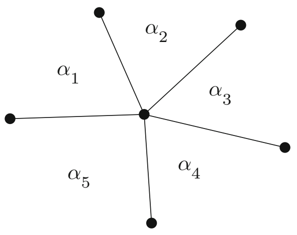


Fig. 3.14 Example of a vertex of degree $d = 5$ and corresponding angles $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$ listed in a clockwise order

Consider the case $k = 2$ of two antennae per sensor and take a vertex u of degree d. We know from Lemma 1 that for $k = 2 \leq d$ antennae, $2(d-2)\pi/d$ is always sufficient and sometimes necessary on the sum of the angles of the antennae at u so that there is a directional antenna from u pointing to all its neighbors. Observe that $2(d-2)\pi/d \leq 6\pi/5$ is always true. Now take an MST with max degree five. Do a preorder traversal that comes back to the starting vertex (any starting vertex will do). For any vertex u arrange the two antennae at u so that there is always a directed edge from u to all its neighbors (if the degree of vertex is 2 you need only one antenna at that vertex). It is now easy to show by following the "underlying" preorder traversal on this tree that the resulting graph is strongly connected.

Consider the case $k = 3$ of three antennae per sensor. First, assume the sum of the three angles is at least $4\pi/5$. Consider an arbitrary vertex u of the MST. We are interested in showing that for this angle there is always a link from u to all its neighbors. If the degree of u is at most three the proof is easy. If the degree is four then by Lemma 1, $2(4-3)\pi/4 = \pi/2$ is sufficient. Finally, if the degree of u is five then again by Lemma 1 then $2(5-3)\pi/5 = 4\pi/5$ is sufficient. Thus, in all cases a sum of $4\pi/5$ is sufficient.

Consider the case $k = 4$ of four antennae per sensor. First, assume that the sum of the four angles is at least $2\pi/5$. Consider an arbitrary vertex, say u, of the MST. If it has degree at most four then clearly four antennae each of angle 0 is sufficient. If it has degree five then an angle between two adjacent neighbors of u, say u_0, u_1, must be $\leq 2\pi/5$ (see left picture in Fig. 3.15). Therefore use the angle $2\pi/5$ to cover both of these sensors and the remaining three antennae (each of spread 0) to reach from u the remaining three neighbors.

Finally, for the case $k = 5$ of five antennae per sensor the result follows immediately from the fact that the underlying MST has maximum degree five. This proves the theorem.

In fact, the result of Theorem 3 can be used to provide better trade-offs on the maximum antennae range and sum of angles. We mention without proof that as consequence of Lemma 1 and Theorem 3 we can construct Table 3.1, which shows trade-offs on the number, max range, max angle, and sum of angles of k antennae,

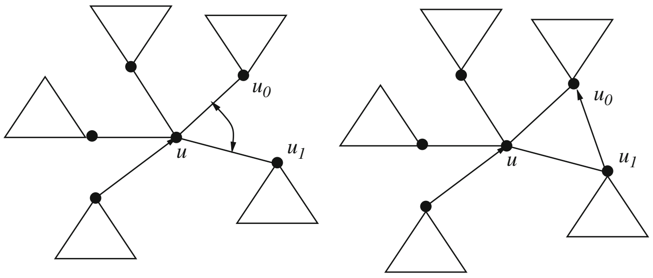


Fig. 3.15 Orienting antennae around u

Table 3.1 Trade-offs on the number, max range, max angle, and sum of angles of k antennae being used by a sensor

Number	Max range	Max angle	Sum of angles
1	2	$\varphi \geq 0$	0
1	$\sqrt{3}$	$\varphi \geq \pi$	π
1	$\sqrt{2}$	$\varphi \geq 4\pi/3$	$4\pi/3$
1	$2\sin(\pi/5)$	$\varphi \geq 3\pi/2$	$3\pi/2$
1	1	$\varphi \geq 8\pi/5$	$8\pi/5$
2	$\sqrt{3}$	$\varphi \geq 0$	0
2	$\sqrt{2}$	$\varphi \geq 2\pi/3$	$2\pi/3$
2	$2\sin(\pi/5)$	$\varphi \geq 2\pi/3$	π
2	1	$\varphi \geq 4\pi/5$	$6\pi/5$
3	$\sqrt{2}$	$\varphi \geq 0$	0
3	$2\sin(\pi/5)$	$\varphi \geq \pi/2$	$\pi/2$
3	1	$\varphi \geq 2\pi/5$	$4\pi/5$
4	$\sqrt{2}$	$\varphi \geq 0$	0
4	1	$\varphi \geq 2\pi/5$	$2\pi/5$
5	1	$\varphi \geq 0$	0

being used per sensor for the problem of converting networks of omnidirectional sensors into strongly connected networks of sensors.

3.4.1 Further Questions and Open Problems

There are two versions of the antennae orientation problem that have been studied. In the first, we are concerned with minimizing the max sensor angle. In the second, discussed in this section, we looked at minimizing the sum of the angles. Aside from the results outlined in Table 3.1, nothing better is known concerning the optimality of the sum of the sensor angles for a given sensor range. Interesting open questions for these problems arise when one has to "respect" a given underlying network of sensors. One such problem is investigated in the next section.

3.5 Orienting Planar Spanners

All the constructions previously considered relied on orienting antennae of a set S of sensors in the plane. Regardless of the construction, the underlying structure connecting the sensors was always an MST on S. However, there are instances where an MST on the point set may not be available because of locality restrictions on the sensors. This is, for example, the case when the spanner results from application of a local planarizing algorithm on a unit disk graph (e.g., see [7, 26]). Thus, in this section we consider the case whereby the underlying network is a given planar spanner on the set S. In particular we have the following problem.

Let $G(V, E, F)$ be a planar geometric graph with V as set of vertices, E as set of edges and F as set of faces. We would like to orient edges in E so that the resulting digraph is strongly connected as well as study trade-offs between the number of directed edges and stretch factor of the resulting graphs.

A trivial algorithm is to orient each edge in E in both directions. In this case, the number of directed edges is $2|E|$ and the stretch factor is 1. Is it possible to orient some edges in only one direction so that the resulting digraph is strongly connected with bounded stretch factor? The answer is yes and an intuitive idea of our approach is based on a c-coloring of faces in F, for some integer c. The idea of using face coloring was used in [40] to construct directed cycles. Intuitively we give directions to edges depending on the color of their incident faces.

3.5.1 Basic Construction

Theorem 8 (Kranakis et al. [25]) *Let $G(V, E, F)$ be a planar geometric graph having no cut edges. Suppose G has a face c-coloring for some integer c. There exists a strongly connected orientation* $\mathbf{G}$ *with at most*

$$\left(2 - \frac{4c-6}{c(c-1)}\right) \cdot |E| \tag{3.1}$$

directed edges, so that its stretch factor is $\Phi(G) - 1$, where $\Phi(G)$ is the largest degree of a face of G.

Before giving the proof, we introduce some useful ideas and results that will be required. Consider a planar geometric graph $G(V, E, F)$ and a face c-coloring C of G with colors $\{1, 2, \ldots, c\}$.

Definition 10 Let $\mathbf{G}$ be the orientation resulting from giving two opposite directions to each edge in E.

Definition 11 For each directed edge (u, v), we define L_{uv} as the face which is incident to $\{u, v\}$ on the left of (u, v), and similarly R_{uv} as the face which is incident to $\{u, v\}$ on the right of (u, v).

Observe that for given embedding of G, L_{uv}, and R_{uv} are well defined. Since G has no cut edges, $L_{uv} \neq R_{uv}$. This will be always assumed in the proofs below without specifically recalling it again. We classify directed edges according to the colors of their incident faces.

Definition 12 Let $E_{(i,j)}$ be the set of directed edges (u, v) in $\mathbf{G}$ such that $C(L_{uv}) = i$ and $C(R_{uv}) = j$.

It is easy to see that each directed edge is exactly in one such set. Hence, the following lemma is evident and can be given without proof.

Lemma 2 *For any face c-coloring of a planar geometric graph G,*

$$\sum_{i=1}^{c} \sum_{j=1, j \neq i}^{c} |E_{(i,j)}| = 2|E|$$

Definition 13 For any of $c(c-1)$ ordered pairs of two different colors a and b of the coloring C, we define the digraph $D(G; a, b)$ as follows: The vertex set of the digraph D is V and the edge set of D is

$$\bigcup_{i \in [1,c]] \setminus \{b\},\, j \in [1,c] \setminus \{a\}} E_{(i,j)}$$

Along with this definition, for $i \neq b$, $j \neq a$, and $i \neq j$, we say that $E_{(i,j)}$ is in $D(G; a, b)$. Next consider the following characteristic function

$$\chi_{a,b}(E_{(i,j)}) = \begin{cases} 1 & \text{if } E_{(i,j)} \text{ is in } D(G; a, b), \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

We claim that every set $E_{(i,j)}$ is in exactly $c^2 - 3c + 3$ different digraphs $D(G; a, b)$ for some $a \neq b$.

Lemma 3 *For any face c-coloring of a planar geometric graph G,*

$$\sum_{a=1}^{c} \sum_{b=1, b \neq a}^{c} \chi_{a,b}(E_{(i,j)}) = c^2 - 3c + 3.$$

Proof Let $i, j \in [1, c]$, $i \neq j$ be fixed. For any two distinct colors a and b of the c-coloring of G, $\chi_{a,b}(E_{(i,j)}) = 1$ only if either $i = a$ or $j = b$, or i and j are different from a and b. There are $(c-1) + (c-2) + (c-2)(c-3)$ such colorings. The lemma follows by simple counting.

The following lemma gives a key property of the digraph $D(G; a, b)$.

Lemma 4 *Given a face c-coloring of a planar geometric graph G with no cut edges, and the corresponding digraph $D(G; a, b)$. Every face of $D(G; a, b)$, which has color a, constitutes a counterclockwise-directed cycle, and every face which has color b constitutes a clockwise-directed cycle. All edges on such cycles are unidirectional. Moreover, each edge of $D(G; a, b)$ incident to faces having colors different from either a or b is bidirectional.*

Proof Let G be a planar geometric graph with a face c-coloring C with colors a, b, and $c-2$ other colors. Consider $D(G; a, b)$. The sets $E_{(a,x)}$ are in $D(G; a, b)$ for each color $x \neq a$. Let f be a face and let $\{u, v\}$ be an edge of f so that $L_{uv} = f$. Let f' be the other face incident to $\{u, v\}$; hence $R_{uv} = f'$. Since G has no cut edges, $f \neq f'$, and, since $C(f') \neq a$, the directed edge $(u, v) \in \bigcup_{x \neq a} E_{(a,x)}$ and hence the edge (u, v) is in $D(G; a, b)$. Since $\{u, v\}$ was an arbitrary edge of f, f will induce a counterclockwise cycle in $D(G; a, b)$ (see Fig. 3.16). The fact that every

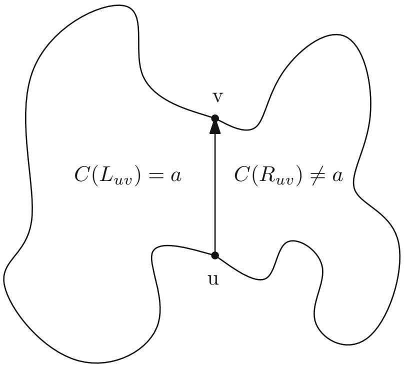


Fig. 3.16 (u, v) is in $D(G; a, b)$ if $C(L_{uv}) = a$ and therefore the edges in the face L_{uv} form a counterclockwise-directed cycle in $D(: G, a, b)$

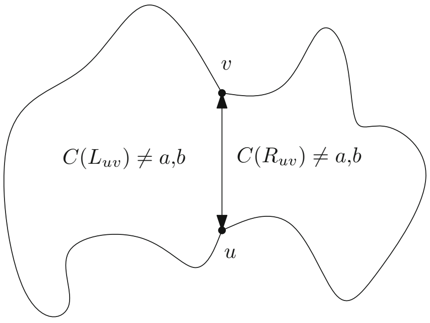


Fig. 3.17 A bidirectional edge is in $D(G; a, b)$ if its incident faces have color different than a and b

face which has color b induces a clockwise cycle in $D(G; a, b)$ is similar. Finally consider an edge $\{u, v\}$ such that $C(L_{uv}) \neq a, b$ and $C(R_{uv}) \neq a, b$ (see Fig. 3.17). Hence $(u, v) \in E_{(c,d)}$ which is in $D(G; a, b)$ and similarly $(v, u) \in E_{(d,c)}$ which is also in $D(G; a, b)$. This proves the lemma.

We are ready to prove Theorem 8.

Proof (Theorem 8) Let G be a planar geometric graph having no cut edges. Let C be a face c-coloring of G with colors a, b and other $c - 2$ colors. Suppose colors a and b are such that the corresponding digraph $D(G; a, b)$ has the minimum number of directed edges. Consider $\overline{D}$ the average number of directed edges in all digraphs arising from C. Thus,

$$\overline{D} = \frac{1}{c(c-1)} \sum_{a=1}^{c} \sum_{b=1, b \neq a}^{c} \|D(G; a, b)\|, \text{ where}$$

$$||D(G; a, b)|| = \sum_{i=1}^{c} \sum_{j=1, j \neq i}^{c} \chi_{a,b}(E_{(i,j)})|E_{(i,j)}|$$

By Lemma 2 and Lemma 3,

$$\begin{aligned}
\overline{D} &= \frac{1}{c(c-1)} \sum_{a=1}^{c} \sum_{b=1,b\neq a}^{c} \sum_{i=1}^{c} \sum_{j=1,j\neq i}^{c} \chi_{a,b}(E_{(i,j)})|E_{(i,j)}| \\
&= \frac{1}{c(c-1)} \sum_{i=1}^{c} \sum_{j=1,j\neq i}^{c} (c^2-3c+3)|E_{(i,j)}| \\
&= \frac{2(c^2-3c+3)}{c(c-1)}|E| \\
&= \left(2 - \frac{4c-6}{c(c-1)}\right) \cdot |E|
\end{aligned}$$

Hence $D(G; a, b)$ has at most the desired number of directed edges.

To prove the strong connectivity of $D(G; a, b)$, consider any path, say $u = u_0, u_1, \ldots, u_n = v$, in the graph G from u to v. We prove that there exists a directed path from u to v in $D(G; a, b)$. It is enough to prove that for all i there is always a directed path from u_i to u_{i+1} for any edge $\{u_i, u_{i+1}\}$ of the above path. We distinguish several cases.

- Case 1. $C(L_{u_i u_{i+1}}) = a$. Then $(u_i, u_{i+1}) \in E_{(a,\omega)}$ where $\omega = C(R_{u_{i}u_{i+1}})$. Since $E_{(a,\omega)}$ is in $D(G; a, b)$, the edge (u_i, u_{i+1}) is in $D(G; a, b)$. Moreover, the stretch factor of $\{u_i, u_{i+1}\}$ is one.
- Case 2. $C(L_{u_i u_{i+1}}) = b$. Hence, (u_i, u_{i+1}) is not in $D(G; a, b)$. However, by Lemma 4, the face $L_{u_i u_{i+1}} = R_{u_{i+1} u_i}$ constitutes a clockwise-directed cycle and therefore, a directed path from u_i to u_{i+1}. It is easy to see that the stretch factor of $\{u_{i,u_{i+1}}\}$ is not more than the size of the face $L_{u_i u_{i+1}}$ minus one, which is at most $\Phi(G) - 1$.
- Case 3. $C(L_{u_i u_{i+1}}) \neq a, b$. Suppose $C(L_{u_i u_{i+1}}) = c$. Three cases can occur.
  - $C(R_{u_i u_{i+1}}) = a$. Hence, (u_i, u_{i+1}) is not in $D(G; a, b)$. However, by Lemma 4, there exists a counterclockwise-directed cycle around face $R_{u_i u_{i+1}} = L_{u_{i+1} u_i}$, and consequently a directed path from u_i to u_{i+1}. The stretch factor is at most the size of face $R_{u_i u_{i+1}}$ minus one, which is at most $\Phi(G) - 1$.
  - $C(R_{u_i u_{i+1}}) = b$. By Lemma 4, there exists a clockwise-directed cycle around face $R_{u_i u_{i+1}}$. This cycle contains $(u_i u_{i+1})$, and in addition the stretch factor of $\{u_i, u_{i+1}\}$ is one.
  - $C(R_{u_i u_{i+1}}) = d \neq a, b, c$. By construction, $D(G; a, b)$ has both edges (u_i, u_{i+1}) and (u_{i+1}, u_i). Again, the stretch factor of $\{u_i, u_{i+1}\}$ is one.

This proves the theorem.

As indicated in Theorem 8 the number of directed edges in the strongly oriented graph depends on the number c of colors according to the formula $\left(2 - \frac{4c-6}{c(c-1)}\right) \cdot |E|$. Thus, for specific values of c we have the following table of values:

c	3	4	5	6	7
$2 - (4c-6)/c(c-1)$	1	7/6	13/10	7/5	31/21

Regarding the complexity of the algorithm, this depends on the number c of colors being used. For example, computing a four coloring can be done in $O(n^2)$ [35]. Finding the digraph with minimum number of directed edges among the 12 possible digraphs can be done in linear time. Therefore, for $c = 4$ it can be computed in $O(n^2)$. For $c = 5$ a five coloring can be found in linear time $O(n)$. For the case of geometric planar subgraphs of unit disk graphs and location aware nodes there is a local seven coloring (see [9]). For more information on colorings the reader is advised to look at [20].

3.5.1.1 Further Questions and Open Problems

Observe that it is required that the underlying geometric graph in Theorem 8 does not have any cut edges. Although it is well known how to construct planar graphs with no cut edges starting from a set of points (e.g., Delaunay triangulation) there are no known constructions in the literature of "local" spanners from UDGs which also guarantee planarity, network connectivity, and no cut edges at the same time. Constructions of spanners obtained by deleting edges from the original graph can be found in Cheriyan et al. [8] and Dong et al. [11] but the algorithms are not local and the spanners not planar. Similarly, existing constructions for augmenting (i.e., adding edges) graphs into spanners with no cut edges (see Rappaport [34], Abellanas et al. [1], Rutter et al. [36]) are not local algorithms and the resulting spanners not planar.

3.6 Conclusion

We considered the problem of converting a planar (undirected) graph constructed using omnidirectional antennae into a planar-directed graph constructed using directional antennae. In our approach we considered trade-offs on the number of antennae, antennae angle, sum of angles of antennae, stretch factor, lower and upper bounds on the feasibility of achieving connectivity. In addition to closing several existing gaps between upper and lower bounds for the algorithms we proposed there remain several open problems concerning topology control whose solution can help to illuminate the relation between networks of omnidirectional and directional antennae. Also of interest is the question of minimizing the amount of energy required to maintain connectivity given one or more directional antennae of a given angular spread in replace of a single omnidirectional antennae.

Acknowledgments Many thanks to the anonymous referees for useful suggestions that improved the presentation. Research supported in part by NSERC (Natural Sciences and Engineering Research Council of Canada), MITACS (Mathematics of Information Technology and Complex Systems), and CONACyT (Consejo Nacional de Ciencia y Tecnología) grants.

References

1. M. Abellanas, A. García, F. Hurtado, J. Tejel, and J. Urrutia. Augmenting the connectivity of geometric graphs. *Computational Geometry: Theory and Applications*, 40(3):220–230, 2008.
2. S. Arora and K. Chang. Approximation Schemes for Degree-Restricted MST and Red–Blue Separation Problems. *Algorithmica*, 40(3):189–210, 2004.
3. L. Bao and J. J. Garcia-Luna-Aceves. Transmission scheduling in ad hoc networks with directional antennas. *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, pages 48–58, Atlanta, Georgia, USA, 2002.
4. B. Bhattacharya, Y Hu, E. Kranakis, D. Krizanc, and Q. Shi. Sensor Network Connectivity with Multiple Directional Antennae of a Given Angular Sum. *23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS 2009), May 25–29*, Rome, Italy, 2009.
5. I. Caragiannis, C. Kaklamanis, E. Kranakis, D. Krizanc, and A. Wiese. Communication in Wireless Networks with Directional Antennae. *In proceedings of 20th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'08), June 14–16*, pages 344–351, Munich, Germany, 2008.
6. T. M. Chan. Euclidean bounded-degree spanning tree ratios. *Discrete and Computational Geometry*, 32(2):177–194, 2004.
7. E. Chávez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, and J. Urrutia. Local Construction of Planar Spanners in Unit Disk Graphs with Irregular Transmission Ranges. In *LATIN 2006, LNCS, Vol. 3887*, pages 286–297, 2006.
8. J. Cheriyan, A. Sebö, and Z. Szigeti. An Improved Approximation Algorithm for Minimum Size 2-Edge Connected Spanning Subgraphs. In *Proceedings of the 6th International IPCO Conference on Integer Programming and Combinatorial Optimization*, Vol. 1412, pages 126–136. Springer, Berlin, 1998.
9. J. Czyzowicz, S. Dobrev, H. Gonzalez-Aguilar, R. Kralovic, E. Kranakis, J. Opatrny, L. Stacho, and J. Urrutia. Local 7-Coloring for Planar Subgraphs of Unit Disk Graphs. In *Theory and Applications of Models of Computation: 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008; Proceedings*, vol. 4978, pages 170–181. Springer, Berlin, 2008.
10. S. Dobrev, E. Kranakis, D. Krizanc, O. Morales, J. Opatrny, and L. Stacho. Strong connectivity in sensor networks with given number of directional antennae of bounded angle, 2010. COCOA 2010, to appear.
11. Q. Dong and Y. Bejerano. Building Robust Nomadic Wireless Mesh Networks Using Directional Antennas. In *IEEE INFOCOM 2008. The 27th Conference on Computer Communications*, pages 1624–1632, Phoenix, AZ, USA, 2008.
12. H. Fleischner. The square of every two-connected graph is Hamiltonian. *Journal of Combinatorial Theory*, 3:29–34, 1974.
13. A. Francke and M. Hoffmann. The Euclidean degree-4 minimum spanning tree problem is NP-hard. In *Proceedings of the 25th Annual Symposium on Computational Geometry*, pages 179–188. ACM, New York, NY, 2009.
14. T. Fukunaga. Graph Orientations with Set Connectivity Requirements. In *Proceedings of the 20th International Symposium on Algorithms and Computation*, pages 265–274. Springer, LNCS, Honolulu, Hawaii, 2009.
15. M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar Hamiltonian circuit problem is NP-complete. *SIAM Journal on Computing*, 5:704, 1976.
16. P. Gupta and P. R. Kumar. The capacity of wireless networks. *Information Theory, IEEE Transactions On*, 46(2):388–404, 2000.
17. L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *Network and Distributed System Security Symposium (NDSS)*, pages 131–141. Internet Society, San Diego, California, USA, 2004.
18. H. Imai, K. Kobara, and K. Morozov. On the possibility of key agreement using variable directional antenna. In *Proc. of 1st Joint Workshop on Information Security,(Korea)*. IEICE, 2006.

19. A. Itai, C.H. Papadimitriou, and J.L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11:676, 1982.
20. T. R. Jensen and B. Toft. *Graph Coloring Problems*. Wiley-Interscience, New York, NY, 1996.
21. S. Khuller, B. Raghavachari, and N. Young. Low degree spanning trees of small weight. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, page 421. ACM, Montreal, Quebec, Canada, 1994.
22. S. Khuller, B. Raghavachari, and N. Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, 14(4):305–321, 1995.
23. E. Kranakis, D. Krizanc, and J. Urrutia. Coverage and Connectivity in Networks with Directional Sensors. *proceedings Euro-Par Conference, Pisa, Italy, August*, pages 917–924, Pisa, Italy, 2004.
24. E. Kranakis, D. Krizanc, and E. Williams. Directional versus omnidirectional antennas for energy consumption and k-connectivity of networks of sensors. *Proceedings of OPODIS*, 3544:357–368, 2004.
25. E. Kranakis, O. Morales, and L. Stacho. On orienting planar sensor networks with bounded stretch factor, 2010. Unpublished manuscript.
26. X. Li, G. Calinescu, P Wan, and Y. Wang. Localized Delaunay Triangulation with Application in Ad Hoc Wireless Networks. *IEEE Transactions on Parallel and Distributed Systems*, 14:2003, 2003.
27. X. Lu, F. Wicker, P. Lio, and D. Towsley. Security Estimation Model with Directional Antennas. In *IEEE Military Communications Conference, 2008. MILCOM 2008*, pages 1–6, 2008.
28. C. Monma and S. Suri. Transitions in geometric minimum spanning trees. *Discrete and Computational Geometry*, 8(1):265–293, 1992.
29. C. S. J. A. Nash-Williams. On orientations, connectivity and odd vertex pairings in finite graphs. *Canadian Journal of Mathematics*, 12:555–567, 1960.
30. V. Navda, A. P. Subramanian, K. Dhanasekaran, A. Timm-Giel, and S. Das. Mobisteer: using steerable beam directional antenna for vehicular network access. In *ACM MobiSys*, 2007.
31. R. G. Parker and R. L. Rardin. Guaranteed performance heuristics for the bottleneck traveling salesman problem. *Operations Research Letters*, 2(6):269–272, 1984.
32. A. P. Punnen. Minmax strongly connected subgraphs with node penalties. *Journal of Applied Mathematics and Decision Sciences*, 2:107–111, 2005.
33. R. Ramanathan. On the performance of ad hoc networks with beamforming antennas. *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 95–105, Long Beach, CA, USA, 2001.
34. D. Rappaport. Computing simple circuits from a set of line segments is NP-complete. *SIAM Journal on Computing*, 18(6):1128–1139, 1989.
35. N. Robertson, D. Sanders, P. Seymour, and R. Thomas. The four-colour theorem. *Journal of Combinatorial Theory Series B*, 70(1):2–44, 1997.
36. I. Rutter and A. Wolff. Augmenting the connectivity of planar and geometric graphs. *Electronic Notes in Discrete Mathematics*, 31:53–56, 2008.
37. A. Spyropoulos and C. S. Raghavendra. Energy efficient communications in ad hoc networks using directional antennas. *INFOCOM 2002: Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, New York, NY, USA, 2002.
38. A. Spyropoulos and C. S. Raghavendra. Capacity bounds for ad-hoc networks using directional antennas. *ICC'03: IEEE International Conference on Communications*, Anchorage, Alaska, USA, 2003.
39. S. Yi, Y. Pei, and S. Kalyanaraman. On the capacity improvement of ad hoc wireless networks using directional antennas. *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 108–116, Annapolis, Maryland, USA, 2003.
40. H. Zhang and X. He. On even triangulations of 2-connected embedded graphs. *SIAM Journal on Computing*, 34(3):683–696, 2005.

Chapter 4
Optimal Placement of Ad Hoc Devices Under a VCG-Style Routing Protocol

Peter Widmayer, Luzi Anderegg, Stephan Eidenbenz, and Leon Peeters

Abstract Motivated by a routing protocol with VCG-style payments, we investigate the combinatorial problem of placing new devices in an ad hoc network such that the resulting shortest path transmission costs, defined as sums of squared Euclidean distances, are minimum. For the cases of only one new device and of one communication request with multiple devices with identical transmission ranges, we provide polynomial-time algorithms. On the negative side, we show that even for a single communication request, placing multiple new devices with different transmission ranges is NP-hard. For identical transmission ranges, the placement of multiple new devices is NP-hard under multiple communication requests.

4.1 Introduction

Wireless ad hoc networks promise the functionality of classical networks, without the burden of having to construct and install a fixed network infrastructure. Each wireless device in an ad hoc network has a restricted transmission range, and communication between two devices typically takes place in a multi-hop fashion along intermediate devices. It is far from clear, however, if and why an intermediate device would be willing to sacrifice its own battery power and bandwidth to forward data packets destined for other devices.

Recently, several papers have addressed the issues caused by selfish devices in wireless ad hoc networks. In particular, various routing protocols have been proposed [3, 10] that issue payments to the intermediate wireless devices, so as to compensate them for their energy costs. This compensation follows the marginal contribution principle by Vickrey, Clarke, and Groves (VCG), the key idea of the issued *VCG payments* being to reward a device for the gain in overall benefit that its participation causes (a good overview of VCG mechanisms is provided in [19]).

The VCG nature of the payments guarantees that devices will truthfully report their distances to other devices. In particular, the profit a device makes from the

P. Widmayer (✉)
Institute of Theoretical Computer Science, ETH Zürich, Zürich, Switzerland
e-mail: widmayer@inf.ethz.ch

VCG payments depends on its position in the network. This motivates the main question studied in this chapter: Where should a device position itself in the network so as to maximize its profit from the VCG payments? Although inspired by a game-theoretic setting, this is a purely combinatorial question that we study in the more general setting of several devices to be positioned for several communication requests, under a transmission cost model that is quadratic in Euclidean distance.

4.1.1 Model and Notation

More formally, we model the above setting as a directed graph $G = (V, E)$, with the vertex set $V = \{1, \dots, n\}$ representing the set of incumbent wireless devices. Each vertex is embedded in the plane, and its coordinates are specified by a placement function $p : V \to \mathbb{R}^2$. We base our cost function on the Euclidean distance measure, with $|uv|$ denoting the distance between two devices u and v, and also writing $|xx'|$ for the Euclidean distance between two points x, x' in the plane. We assume that the distance between any two device positions can be computed in constant time. The transmission ranges of the devices are modeled by a transmission range function $r : V \to \mathbb{R}_+$, specifying the maximal distance $r(u)$ from device u at which another device can still receive a signal from u via direct communication.

The edge set E of size m contains a directed edge (u, v) whenever device v lies within the transmission range of device u, that is, if and only if $|uv| \leq r(u)$. The cost $c(u, v)$ of a directed edge (u, v) reflects the energy requirement for transmitting a unit size data packet along the edge. Following the most common theoretical models of power attenuation, the cost is taken proportional to the squared Euclidean distance as $c(u, v) = \gamma|uv|^2$, with γ some constant; powers other than 2 of the Euclidean distance are easily handled along the same lines. For convenience, we set the cost of all non-edges $(u, v) \notin E$ to $c(u, v) = \infty$.

The network needs to accommodate a number of communication requests between devices, which we model by a commodity set $K = \{(s_1, t_1), \dots, (s_k, t_k)\}$, with s_i and t_i being the ith source device and destination device, respectively. Each communication request is for a single unit size packet, and no two commodities share both the source and destination device. Hence, k can be as large as $n \times (n - 1)$. If there is only one commodity, then we denote the source by s and the destination by t. We refer to a tuple of the form (V, E, K, p, r, c) as a transmission graph T, where E is a function of p and r and c is a function of p and γ.

The cost of a path $P = (v_1, \dots, v_j)$ is $c(P) = \sum_{i=1}^{j-1} c(v_i, v_{i+1})$, as usual. By $SP_T(s, t)$ we denote a shortest directed path in the transmission graph T from s to t with respect to the edge costs c, i.e., a path P from s to t for which $c(P)$ is smallest. Further, $SP_T^{-u}(s, t)$ denotes the length of a shortest $s - t$-path not using the vertex u, and $SP_T^{-U}(s, t)$ the length of a shortest $s - t$-path not using any vertex in the set of vertices $U \subset V$. By $c(T) = \sum_{i \in K} c(SP_T(s_i, t_i))$ we refer to the total path costs over all commodities. We will assume that every commodity in K is connected by a path of finite cost.

For finding a shortest path from a source vertex s to a destination vertex t in a transmission graph T, the following incentive-compatible ad hoc VCG protocol has been proposed [3]. First, the protocol basically asks the vertices for their positions and mutual distances during a flooding broadcast phase. Using this information, the protocol computes the edge costs $c(u, v)$ and a shortest path $SP_T(s, t)$. Finally, it pays each vertex $u \in SP_T(s, t)$ an amount $c\left(SP_T^{-u}(s, t)\right) - c(SP_T(s, t)) + c(u, v)$, where (u, v) is the outgoing edge of u in $SP_T(s, t)$. Because of the VCG nature of the payments, vertices have no incentive to lie, and hence the computed $c(u, v)$ can be assumed to equal the true transmission costs. Thus, a vertex u gains a profit of $c\left(SP_T^{-u}(s, t)\right) - c(SP_T(s, t))$. This principle extends to the case where a selfish agent controls a set of devices U and the mechanism knows about this fact and gains a profit of $c\left(SP_T^{-U}(s, t)\right) - c(SP_T(s, t))$.

4.1.2 The Device Placement Problem

Inspired by the ad hoc VCG protocol, this chapter takes the perspective of a profit maximizing selfish agent that enters an existing transmission graph T with a set $\Delta V = \{n + 1, \ldots, n + \Delta n\}$ of Δn new devices, each with a maximal transmission range $r(v)$, $v \in \Delta V$. Assuming that the communication requests for the near future are known, the agent's goal is to determine positions for its Δn devices such that the profit from the resulting VCG payments is maximum. Denoting by T' the new transmission graph including the new devices ΔV at their chosen positions, the objective function is defined as

$$\text{maximize} \sum_{i=1}^{k} (c(SP_T(s_i, t_i)) - c(SP_{T'}(s_i, t_i))) = c(T) - c(T') \tag{4.1}$$

Since the first term in (4.1) is independent of the positions of the devices in ΔV, the problem is equivalent to

$$\text{minimize } c(T') \tag{4.2}$$

More formally, the device placement problem is stated as follows:

Problem: Device Placement.
Instance: An instance $I = (T, \Delta n, r')$ consists of a transmission graph $T = (V, E, K, p, r, c)$, a positive integer Δn, and a maximal transmission range $r'(v)$ for each additional device $v \in \Delta V$.
Question: Find a placement for the Δn additional devices such that the difference $c(T) - c(T')$ is maximum, where T' is the transmission graph after the placement of the additional devices.

Thus, besides the game-theoretic motivation, the resulting *device placement problem* can also be defined as an optimization problem without any game-theoretic flavor: place Δn additional devices such that the shortest paths in the resulting transmission graph T' are of minimum length.

We investigate the algorithmic complexity of the device placement problem for $\Delta n = 1$ as well as for general Δn and for $k = 1$ communication request as well as for general k. Depending on the form of the maximal transmission ranges of the additional devices, we study two problem variants: with *identical* new devices that each have the same transmission range $r(n+1) = r(n+2) = \cdots = r(n+\Delta n)$ and with *individual* new devices, each having their own transmission range $r(v)$, $v \in \Delta V$. Clearly, the two problem variants do not differ for a single additional device. Hence, we simply refer to this case as the single device placement problem.

We provide a polynomial-time algorithm for the problem of placing Δn new devices with identical transmission ranges under a single communication request. To arrive at this result, we first study the case with a single communication request and a single new device, analyze its geometric structure, and propose geometric objects that capture this structure. We further present a polynomial-time algorithm for optimally placing a single new device under k communication requests.

Furthermore, we show that if the wireless devices have individual, potentially different transmission ranges, the placement problem is NP-hard already for a single communication request. The problem of placing Δn new wireless devices with identical transmission ranges for a number k of communication requests is NP-hard, where Δn and k are part of the input.

4.1.3 Related Work

Network upgrade problems where an existing network has to be extended such that the resulting network exhibits certain properties are classical optimization problems. Several variants of these problems have been considered, and the work closest to ours, although still quite different, is the thesis by Krumke [18]. Given a graph and a function specifying the cost of shortening an edge, he investigated how to determine an optimal strategy to minimize the total weight of a minimum spanning tree within a budget restriction.

Our approach introduces concepts and methods from computational geometry into the multi-hop wireless networking domain. A similar combination of techniques has been applied to show hardness results for scheduling on the medium access control layer in an ad hoc network setting [13, 14].

The idea of our approach in this chapter is in a similar vein to the work on network creation games in [11]. The main goal there is to explain the structure of networks constructed by independent selfish agents from a game-theoretic point of view. Different authors [2, 5–7, 15, 20] continued this line of study in related network creation models, including a geometric model [16] similar in spirit to ours.

4.2 Placing Multiple Identical Devices for a Single Commodity

This section studies the basic geometric structure of the identical device placement problem for a single commodity. We first characterize the optimal position of one additional device. Next, we use that characterization to construct an algorithm for optimally placing an additional device and extend that algorithm to compute the optimal positions of multiple identical devices.

4.2.1 The Optimal Position of a Single Additional Device

Suppose the transmission graph consists of only two devices u and v that wish to communicate, and we are interested in the best position for an additional device v'. Let the impact $F_{uv}(v')$ of the additional device v' be the difference between the cost of the direct communication from u to v and the cost of the communication from u to v via the additional device v'. That is, $F_{uv}(v') = c(u, v) - (c(u, v') + c(v', v))$. Note that the impact may be negative. Figure 4.1 illustrates the following observation relating the impact $F_{uv}(v')$ to the position of v'.

Observation 1 *The impact of an additional device v' between devices u and v is equal to $F_{uv}(v') = c(u, v)/2 - 2\gamma \cdot |v', M_{uv}|^2$, where M_{uv} is the middle point of the line segment from u to v.*

Observation 1 implies that device positions with the same impact lie on a circle with center M_{uv}, with the maximum impact achieved at M_{uv}. From there the impact decreases quadratically in each direction, and it is equal to zero for positions on a circle with center M_{uv} and radius $|uv|/2$.

Next, we include a single source–destination pair (s, t) into the impact function. To that end, we define the impact $F^{st}_{uv}(v')$ of an additional device v' on a device pair (u, v) with respect to the single source–destination pair (s, t) as the difference between the shortest $s - t$-path length without v' and with v' and (u, v'), (v', v) as a *mandatory* partial path.

Observation 2 *The impact of an additional device v' for a device pair (u, v) and a source–destination pair (s, t) is*

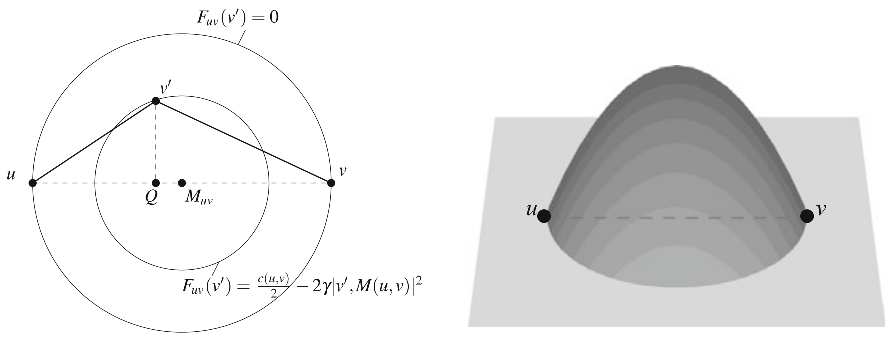


Fig. 4.1 Points with same impact $F_{uv}(v')$ are on circles around $M(u, v)$ (*left*), and the corresponding graph of $F_{uv}(v')$ in $\mathbb{R}^3$

$$F_{uv}^{st}(v') = c(SP_T(s,t)) - c(SP_T(s,u)) - 2\gamma \cdot |v', M_{uv}|^2 - c(u,v)/2 - c(SP_T(v,t))$$

Proof $F_{uv}^{st}(v') = c(SP_T(s,t)) - [c(SP_T(s,u)) + c(u,v') + c(v',v) + c(SP_T(v,t))]$, so the observation follows by using Observation 1. □

Note that the impact is defined for every pair of devices $u, v \in V$, and that it can be negative. An additional device induces a shortest path along (u, v', v) if its impact is positive. The impact is again equal for all positions with the same distance to M_{uv}, and the maximum impact is achieved at position M_{uv}. Observe that the circle with positions of zero impact does not necessarily go through the positions of devices u and v. Indeed, if u and v are not on a shortest path before inserting the additional device, then the circle with positions of zero impact has a smaller radius than $|M_{uv}, u|$.

Some positions with positive impact may be unreachable and hence useless due to small maximal transmission ranges of both the additional device and the existing devices. Thus, we define the profit region PR_{uv}^{st} as the set of positions for an additional device v' where $F_{uv}^{st}(v')$ is positive, u can reach v', and v' can reach v, given the maximal transmission ranges. Geometrically, a profit region PR_{uv}^{st} is the intersection of three disks: the disk around M_{uv} where $F_{uv}^{st}(v') \geq 0$, the disk with center u and radius $r(u)$, and the disk with center v and radius $r(v')$ (see Fig. 4.2 for three possible shapes of such an intersection). The boundary of a shape consists of at most four circle segments. We define $G_{uv}^{st}(\cdot)$ to be the function $F_{uv}^{st}(\cdot)$ restricted to the corresponding profit region. That is, $G_{uv}^{st}(v')$ is equal to $F_{uv}^{st}(v')$ for all positions of v' inside PR_{uv}^{st}, and $-\infty$ otherwise.

Assuming that the profit region PR_{uv}^{st} is not empty and that v' is placed between u and v, the position inside PR_{uv}^{st} with minimal distance to the point M_{uv} is the best position for v' because it maximizes $G_{uv}^{st}(v')$. If the maximal transmission ranges of u and v' are large enough, then this is the same as M_{uv} itself. If the maximal transmission range of v' or u is too small, then the best position conceptually moves on the line segment between u and v from M_{uv} toward device u respectively v until it enters the profit region. Such a best position is denoted by $p^*(u, v, (s,t))$, and it can be computed in constant time given the distance between u and v and the maximal transmission ranges $r(u)$ and $r(v')$.

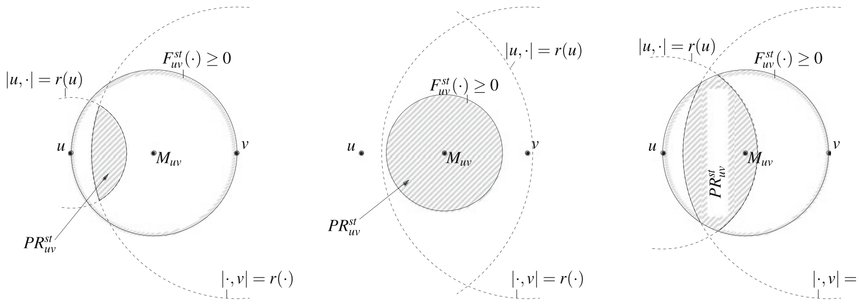


Fig. 4.2 Profit region $PR(u, v, (s,t))$ building an asymmetric lens, a circle, and a shape bounded by four arcs

4.2.2 Multiple Identical Devices

The fact that one additional device reduces the cost between exactly one device pair enables us to state the following geometric formulation for placing a single additional device for a single commodity:

$$\max_{p(v')\in\mathbb{R}^2}\ \max_{u,v\in V} G^{st}_{uv}(v') \tag{4.3}$$

Below, we use this formulation to derive an algorithm for placing multiple identical devices for a single commodity. As a first step, however, we note that Observation 2 and problem formulation (4.2) together induce an algorithm for the simpler problem of optimally placing a single device for a single commodity.

To that end, we define the following *expanded* 2*-layer graph* to encode the restriction that only one additional device is available. The graph has two layers, labeled 0 and 1, each containing a copy of the transmission graph. We add an edge from each vertex $(u, 0)$ on layer 0 to each vertex $(v, 1)$ on layer 1, for $u \neq v$. The cost of such an edge is equal to $c(u, p^*) + c(p^*, v)$, the transmission cost between u and v via an additional device at position $p^*(u, v, (s, t))$. For simplicity, we exclude edges with infinite cost. In this graph, we then search a shortest path from vertex $(s, 0)$ to vertex $(t, 0)$ and another one from $(s, 0)$ to vertex $(t, 1)$. By construction, the minimum of these two paths corresponds to optimally placing the additional device.

The above approach can be extended as follows to optimally place Δn *identical* additional devices, instead of only one. In principle, the best positions for $h \leq \Delta n$ additional devices between a fixed pair (u, v) of devices are to distribute the additional devices in equal distances on the segment connecting u and v. However, limited maximal transmission ranges of device u or of the additional devices may make such equal distances impossible. In such a case, the additional devices are distributed as evenly as possible on the feasible part of the segment connecting u and v. Based on this insight, we construct a $(\Delta n + 1)$-layer graph $H = (V_H, E_H)$ with a copy of the transmission graph on each layer. For each layer $h < \Delta n$ and each "higher" layer $h' > h$, we add an edge from each vertex (u, h) to each vertex (v, h'), for $u \neq v$, the cost of which is equal to the transmission cost from u to v via $(h' - h)$ optimally placed additional devices between u and v, as discussed before. Edges with infinite cost are again excluded for simplicity.

Theorem 1 *The multiple identical device placement problem for a single commodity can be solved in time* $O((\Delta n)^2 n^2)$.

Proof We use the $(\Delta n+1)$-layer graph H described above, compute a shortest path between $(s, 0)$ and (t, h) for each $h, 0 \leq h \leq \Delta n$, and output a path with length $\min_{0\leq h\leq \Delta n} c(SP((s, 0), (t, h)))$. Since the cost of each edge $((u, h), (v, h')) \in E_H$ correctly reflects the cost of a subpath from u to v containing exactly $(h' - h)$ optimally placed additional devices between u and v, the correctness of the algorithm follows. The construction of H needs time $O((\Delta n)^2 n^2)$, as there are that many potential edges in the graph. All shortest paths can be found in time $O((\Delta n)^2 n^2)$ using Dijkstra's algorithm to find a shortest path tree rooted at $(s, 0)$. □

4.3 Single Device Placement for Multiple Commodities

With multiple commodities ($k > 1$), the optimal position for a single additional device may be different from the optimal point $p^*(u, v, (s_i, t_i))$ between some existing devices u and v and a specific commodity i. Rather, the best position could be a position where connections between several source–destination pairs use the new device. Unfortunately, the ideas from the previous section do not easily extend to a polynomial-time algorithm for the single device and multiple commodities case. Therefore, we first present a different algorithm for the single device and single commodity case, which has worse running time than the algorithm above, but is extendable to the single device and multiple commodities case.

4.3.1 Single Maximization Diagram Approach

An alternative approach to solve the single device and single commodity case is to directly use the geometric formulation in (4.3). There, the term $\max_{u,v \in V} G_{uv}^{st}(\cdot)$ defines exactly the *upper envelope* of the impact functions $G_{uv}^{st}(\cdot)$, $u, v \in V$, that is, the point-wise maximum of the curves $G_{uv}^{st}(\cdot)$. The *maximization diagram* $\mathcal{M}$ of the impact functions $G_{uv}^{st}(\cdot)$ divides the plane into maximal connected cells, such that within one cell the same function $G_{uv}^{st}(\cdot)$ attains the upper envelope defining maximum (see [1] for a detailed description of maximization diagrams). Figure 4.3 shows the upper envelope of two impact functions and the corresponding maximization diagram. Thus, a cell in $\mathcal{M}$ has a *characterizing* device pair, and for each point in the cell, that device pair yields the maximum impact. Inside a given cell, the optimal position for a new device is defined by the maximum of the concave function $G_{uv}^{st}(\cdot)$ for the characterizing device pair (u, v) and is hence easy to com-

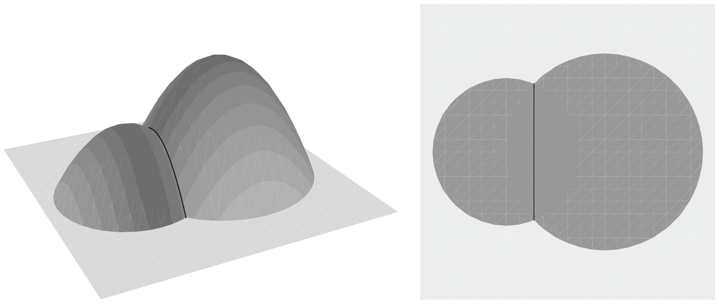


Fig. 4.3 Upper envelope of the impact functions for two device pairs and the corresponding maximization diagram

pute. For a polynomially bounded number of cells, this approach gives rise to a polynomial-time algorithm.

Figure 4.3 illustrates that an edge in $\mathcal{M}$ arises either from the intersection of two impact functions $G^{st}_{uv}(\cdot)$ or from a domain boundary of an impact function. These domain boundaries are circle segments, and the following observation states that an intersection yields a line segment. Thus, the edges of any maximization diagram cell are either line or circle segments.

Observation 3 *The intersection of two impact functions $G^{st}_{u_1v_1}(\cdot)$ and $G^{st}_{u_2v_2}(\cdot)$ is a line.*

Proof Consider the impact functions for device pairs (u_1, v_1) and (u_2, v_2) and an additional device v'. Both impact functions $G^{st}_{u_iv_i}(v')$ are of the form $H_i - 2 \cdot \gamma|v', M_{u_iv_i}|^2$, where the constant H_i depends on the positions of the devices, for $i = 1, 2$. If we set $G^{st}_{u_1v_1}(\cdot) = G^{st}_{u_2v_2}(\cdot)$, then the set of points fulfilling the equation constitutes a line. □

Lemma 1 *Given a two-dimensional maximization diagram cell c, represented by a list of its n_c incident edges, with a characterizing pair (u, v), the optimal position inside c with respect to $G^{st}_{uv}(\cdot)$ can be found in time $O(n_c)$.*

Proof Inside c, the profit of any position is equal to the concave impact function $G^{st}_{uv}(\cdot)$. Hence, the maximum inside c is either attained at the single point where the gradient is equal to zero, if this point lies inside c, or attained somewhere on the boundary of c. For the function $G^{st}_{uv}(\cdot)$, the gradient is zero at position M_{uv}, and if this position is inside c, we are done. Otherwise, we go along the boundary edges of c, where, for a single edge, the maximum is attained at the position with smallest distance to M_{uv}.

Algorithm 1 MaxDiagram(s, t)

```
Output: Optimal position for one additional device for one commodity $(s, t)$.
1: for all device pairs $(u, v)$ do
2:     compute $G^{st}_{uv}(\cdot)$
3: end for
4: compute maximization diagram $\mathcal{M}$ of $\cup_{u,v} G^{st}_{uv}(\cdot)$
5: compute the global optimum over all 2-dimensional max. diagram cells $c \in \mathcal{M}$
```

Testing whether M_{uv} is inside c can be done in time linear in n_c by comparing the position to each edge. The maximum computation for all n_c edges needs linear time as well, since the position on a line segment or circle segment edge with smallest distance to M_{uv} can be determined in constant time. □

Lemma 2 *The single device placement problem for a single commodity can be solved in time $O(n^{4+\varepsilon})$.*

Proof We use Algorithm MaxDiagram that extends the above approach by considering all maximization diagram cells. First, we compute the single source shortest path tree from s and the single destination shortest path tree to t. This can be done in time $O(n \log n + m)$. The for-loop over all device pairs needs time $O(n^2)$, and within one iteration we evaluate G_{uv}^{st} for a device pair (u, v). As a single evaluation can be executed in constant time using the shortest path trees, this step runs in time $O(n^2)$.

It was shown in [1] that the maximization diagram of ℓ partially defined functions in $\mathbb{R}^3$ can be computed in time $O(\ell^{2+\varepsilon'})$, for any $\varepsilon' > 0$. Thus, the maximization diagram $\mathcal{M}$ of the $O(n^2)$ functions $G_{uv}^{st}(\cdot)$ can be computed in time $O(n^{4+\varepsilon})$, and the combinatorial complexity of $\mathcal{M}$ is $O(n^{4+\varepsilon})$ as well. Using Lemma 1 and the fact that each edge is incident to at most two cells, computing the maximum over all two-dimensional cells in $\mathcal{M}$ takes time $O(n^{4+\varepsilon})$. Altogether, the running time is $O(n^{4+\varepsilon})$. □

4.3.2 Multiple Maximization Diagrams Approach

Next, we extend the above approach to the single device placement problem for multiple commodities, by means of the following formulation:

$$\max_{p(v')\in\mathbb{R}^2} \sum_{i=1}^{k} \max_{u,v\in V} G_{uv}^{s_i t_i}(v') \tag{4.4}$$

We first compute the maximization diagram $\mathcal{M}_i$ for each commodity $i \in K$. Now, each point in the plane is part of one cell in each $\mathcal{M}_i$, and that cell determines the characterizing pair for the corresponding commodity i (if it exists). We use this fact to determine the regions in which each point has the same characterizing pair *for each single commodity*. That is, we intersect the cell partitions of all maximization diagrams, giving rise to new (smaller) cells. The result can be visualized as the new, fine-grained partition that arises from superimposing transparent slides containing the more coarse-grained maximization diagram cells. More precisely, all the points in a new cell come, for each maximization diagram, from a single cell, and the new cell is a maximal connected region for which this holds. This construct is known as the *overlay* $\mathcal{O}$ of the cell sets $\mathcal{M}_1, \ldots, \mathcal{M}_k$ (see chapter 2 in [8] for details on its

Algorithm 2 MaxDiagramOverlay(K)

```
Output: Optimal position for one additional device for the commodity set $K$.
1: for all commodities $i \in K$ do
2:     for all device pairs $(u, v)$ do
3:         compute $G_{uv}^{s_i t_i}(\cdot)$
4:     end for
5:     compute maximization diagram $\mathcal{M}_i$ of $\cup_{u,v} G_{uv}^{s_i t_i}(\cdot)$
6: end for
7: compute $\mathcal{O} = overlay(\mathcal{M}_1, \ldots, \mathcal{M}_k)$
8: compute the global optimum over all 2-dimensional overlay cells $c \in O$
```

computation). Then, for every single commodity, the characterizing pair is the same for every point in an overlay cell. Lemma 3 states the complexity of computing the optimal position inside a single overlay cell and Theorem 2 the resulting complexity of the above approach.

Lemma 3 *Given a two-dimensional overlay cell c with a (possibly empty) characterizing pair (u_i, v_i) for each commodity $i \in K$, and represented by a list of its n_c incident edges, the optimal position inside c with respect to the profit $\sum_i G_{uv}^{s_i t_i}(\cdot)$ can be found in time $O(n_c)$.*

Proof The position for which $\sum_i G_{uv}^{s_i t_i}(\cdot)$ attains the maximum is the optimal position inside c. As the functions $G_{uv}^{s_i t_i}(\cdot)$ are concave inside c for all characterizing pairs, and for all commodities $i \in K$, the sum over these functions is concave as well. As in Lemma 1, the maximum of the resulting concave function is either attained at the single point where the gradient is zero or on the boundary of c. Here, the single point where the gradient is zero evaluates to the center of mass of the positions $M_{u_i v_i}$. The remainder of the proof is the same as in the proof of Lemma 1. □

Theorem 2 *The single device placement problem for k commodities can be solved in time $O(k^2 n^{8+2\varepsilon} \log(kn^{4+\varepsilon}))$.*

Proof We use Algorithm MaxDiagramOverlay that summarizes the above described approach. The nested for-loop needs time $O(kn^{4+\varepsilon})$ as we compute a maximization diagram for each of k commodities. The overlay of two sets of planar geometric objects with combinatorial complexities ℓ' and ℓ'' can be computed in time $O(\ell \log(\ell' + \ell''))$ where ℓ is the combinatorial complexity of the resulting overlay (see chapter 2 in [8]). As the combinatorial complexity of the overlay is $O((kn^{4+\varepsilon})^2)$, it can be constructed in time $O(k^2 n^{8+2\varepsilon} \log(kn^{4+\varepsilon}))$. Using Lemma 3 and the fact that each edge is incident to at most two cells, the running time for computing the global optimum over all overlay cells is in $O(k^2 n^{8+\varepsilon})$. Thus, the computation of the overlay dominates the overall running time of the algorithm. □

4.4 Placing Multiple Individual Devices for a Single Commodity

For individual devices that each have a specific maximal transmission range, we have to specify exactly which additional device is placed at which position. The decision version of the corresponding problem is defined as follows.

Problem: Individual Device Placement.
Instance: An instance $I = (T, \Delta n, r, Z')$ consists of a transmission graph $T = (V, E, K, p, r, c)$, a positive integer Δn, an individual maximal transmission range $r(v)$ for each additional device $v \in \Delta V$, and a positive number Z'.

Question: Is there a placement for the Δn additional devices such that $c(T') \leq Z'$, where T' is the transmission graph after the placement of the additional devices?

We now prove that this problem is NP-hard already for a single commodity.

Theorem 3 *Individual Device Placement is NP-hard for a single commodity.*

Proof The proof is by a reduction from Partition (SP12 in [12]). In Partition, we are given a set $A = \{a_1, \ldots, a_{|A|}\}$ of positive integer numbers. The goal is to decide whether there is a subset $A' \subseteq A$ such that $\sum_{a_i \in A'} a_i = B/2$, where $B = \sum_{a_i \in A} a_i$. We construct a device placement instance consisting of three devices $\{1, \ldots, 3\}$, placed equidistant on a line, with device 1 at position $\langle 0 \rangle$, device 2 at position $\langle B/2 + 1 \rangle$, and device 3 at position $\langle B + 2 \rangle$. The maximal transmission ranges of these devices are set to 1. Further, the device pair $(1, 3)$ constitutes the single commodity. The number Δn of additional devices is set to $|A|$, and the maximal transmission range $r(u)$ is set to a_{v-n}, for $v = n + 1, \ldots, n + \Delta n$. Finally, we set Z' to $2 + \sum_{a_i \in A} a_i^2$. A solution of the partition problem immediately gives a solution for the device placement problem: we place the devices with index in A' one after another on the line segment between devices 1 and 2, starting at distance 1 from device 1, such that their maximal transmission ranges are just exactly large enough to reach the next device. The remaining devices in $A \setminus A'$ are placed between devices 2 and 3 accordingly. The total cost of the path now equals $1 + \sum_{a_i \in A'} a_i^2 + 1 + \sum_{a_j \in A \setminus A'} a_j^2$, which is $B + 2$. If the partition problem has no solution, then no placement of the devices connects devices 1 and 2 and devices 2 and 3, and hence the total shortest path cost of infinity cannot be avoided. □

4.5 Placing Multiple Devices for Multiple Commodities

We show that the problem of placing multiple new devices for multiple commodities is NP-hard, even if all new devices have the same transmission range. The decision version of this problem is stated as follows:

Problem: Identical Device Placement.
Instance: An instance $I = (T, \Delta n, r, Z)$ consists of a transmission graph $T = (V, E, K, p, r, c)$, a positive integer Δn, an identical maximal transmission range $r(v)$ for each additional device $v \in \Delta V$, and a positive number Z.
Question: Is there a placement for the Δn additional devices such that the difference $c(T) - c(T') \geq Z$, where T' is the transmission graph after the placement of the additional devices?

Theorem 4 *Identical Device Placement is NP-hard.*

Proof Since the proof is quite lengthy and technical, let us first sketch its main idea. We reduce a special version of planar Exact Cover By 3-Sets (X3C) to Identical Device Placement. In an X3C instance $I(U, S, b)$ we are given a set U of $3b$

elements, a collection of 3-element subsets $S = \{S_1, \ldots, S_{|S|}\}$ of U, and a budget b. We are looking for a subcollection of size b from S whose union is U. We use the restricted version of X3C where the corresponding bipartite graph with elements on one side and subsets on the other is planar and each element appears in either two or three sets [9], and denote it by X3C-3.

The idea of the reduction is to introduce a device for every element and every set from the X3C-3 instance. For embedding these devices in the plane, we limit ourselves to instances of X3C-3 whose graph can be augmented (by adding edges) to become triconnected while staying 3-planar. We call this restricted problem *augmentable* X3C-3 or simply ++X3C-3. We then use a result by Kant [17] for drawing a triconnected 3-planar graph with horizontal and vertical edge segments on a grid. Every element device forms one source–destination pair with an additional global destination device.

Further, we show that one can construct a *chain* consisting of a polynomial number of devices at and between any two points x, x', such that the cost of the shortest path between the devices at x and x' is bounded by their distance. Using such chains, we ensure that the only possible paths between an element device and the global destination device go through the set devices the element is a member of. The cost of these paths is the same for all source–destination pairs. Moreover, the placement of an additional device within a chain yields only a small profit. Indeed, only one position induces a large improvement between each set device and the global destination device. Hence, the number of reasonable positions for the additional devices is limited to the number of subsets in S, and there is a one-to-one correspondence between such a position and a subset. This completes the overview of the proof.

Let us now go into the technical details of the proof, by first mentioning the details of the graph drawing result that we use. Recall that a graph is triconnected if no removal of two vertices disconnects the graph. In a 3-planar graph, any vertex has degree at most 3. Any triconnected 3-planar graph is, hence, 3-regular. □

Lemma 4 (Kant [17]) *A triconnected 3-planar graph with n vertices has a planar drawing with horizontal and vertical segments on an $\lceil \frac{n}{2} \rceil \times \lceil \frac{n}{2} \rceil$ grid such that every edge has at most one bend.*

NP-hardness of X3C-3 is proved in [9] by a reduction that creates special X3C-3 instances. Each such instance has the property that it can be augmented to a triconnected 3-planar graph by adding extra edges, and these extra edges can of course be deleted again after drawing the augmented graph.

There are four directions to attach an additional line horizontally or vertically at a vertex v, namely left, right, up, and down. A direction is called *free* for a vertex if no incident line segment goes in this direction, and an additional orthogonal straight line can be drawn in this direction to the outside of the bounding box of the drawing without hitting any other vertex.

Observation 4 *The algorithm from [17] to draw a 3-planar triconnected graph guarantees that every vertex v has a free direction for drawing an additional straight* *line segment. Moreover, no two such straight line segments that belong to two different vertices and are orthogonal intersect each other.*

As a last ingredient we show how to place devices along line segments in the plane such that certain properties hold. As our proof does not make use of the constant γ in the definition of the transmission cost, it works for any value of γ, and we will therefore simply choose $\gamma = 1$ and not mention γ any further.

Lemma 5 *Given two points p_1, p_2 in Euclidean distance ℓ, and $z \leq \ell^2$, we can place two devices u, v at p_1, p_2 and further devices on the line segment between p_1 and p_2 such that $c(SP(u, v)) = z$.*

Proof Let $s(p_1, p_2)$ be the line segment between p_1 and p_2. We distinguish three cases. If $z = \ell^2$, then we place device u with $r(u) = \ell$ at position p_1 and v at position p_2, and no other device on $s(p_1, p_2)$. If ℓ^2 is a multiple of z, then we place device u at p_1 and v at p_2, plus devices on $s(p_1, p_2)$ at distance $h \cdot z/\ell$ from p_1, for $h = 1, \ldots, \ell^2/z - 1$. The maximal transmission range of all devices is set to z/ℓ.

Otherwise (see Fig. 4.4), we place device u at p_1 and v at p_2, plus devices on $s(p_1, p_2)$ at distance $h \cdot z/\ell$ from p_1, for $h = 1, \ldots, \lceil \ell^2/z \rceil - 2$. Let $r_\ell = \ell - \lfloor \ell^2/z \rfloor \cdot z/\ell$. A further device is placed on $s(p_1, p_2)$ at distance $\ell - r_\ell + a$ from p_1, where $a = \left(\sqrt{(z/\ell)^2 - r_\ell^2} - z/\ell + r_\ell\right)/2$. The maximal transmission range of the devices is set such that each device reaches the next device on the segment. The shortest path between u and v goes over each device on $s(p_1, p_2)$. The sum of the squared distances is $(\lceil \ell^2/z \rceil - 2)(z/\ell)^2 + (z/\ell + a)^2 + (r_\ell - a)^2$, which equals z.□

For ease of description, $\mathrm{ch}(p_1, p_2, z)$ denotes a *chain* of devices between positions p_1 and p_2, with positions as in the proof above, and a path of cost z. Similarly, $\mathrm{ch}^-(p_1, p_2, z)$ is the same set of devices without the device at position p_1.

Let us now continue the proof of Theorem 4 by describing in detail the reduction from the instances of planar X3C-3 that appear in the corresponding NP-hardness proof in [9]. We make use of the property of these instances that they can be augmented by adding edges to become triconnected 3-planar. Let us call the planar X3C-3 problem for these instances planar ++X3C-3 and note that the reduction in [9] proves planar ++X3C-3 to be NP-hard. Let $I(U, S, b)$ be an instance of planar ++X3C-3. By Lemma 4, we can assume that an orthogonal grid embedding is given, for which we scale all coordinates by a factor $\hat{c}$ to ensure that vertices are sufficiently

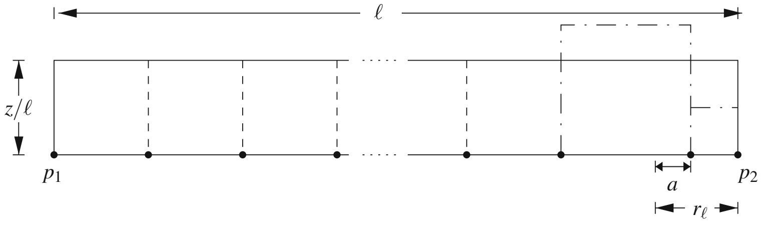


Fig. 4.4 Chain construction

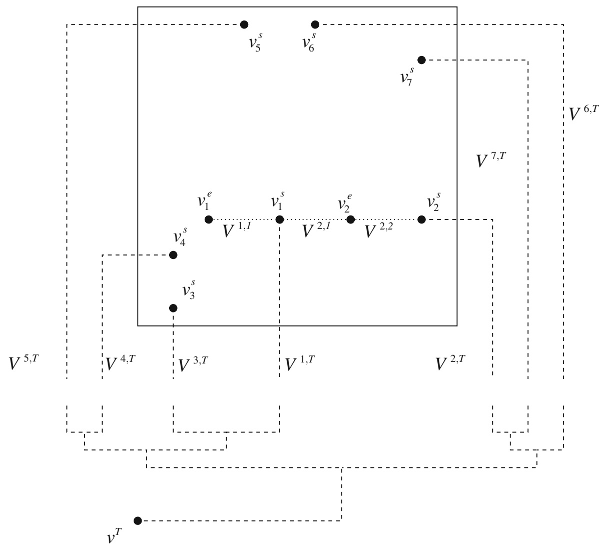


Fig. 4.5 Overview illustration of the reduction where each v_i^e corresponds to an element device and each v_j^s to a set device. v^T is the global destination device, and each $V^{i,j}$ respectively each $V^{j,T}$ refers to an element-set chain respectively a set-destination chain. The gaps on the set-destination chains indicate the potential positions for the additional devices

far away from each other. The factor $\hat{c}$ is chosen to be $4 \times n^3$ for reasons that will become clear later. For now, it is sufficient to note that $\hat{c}$ is polynomially bounded in n. After the scaling, a vertex i has coordinates $\langle i_x, i_y \rangle$. Also by Lemma 4, each edge (i, j) between vertex i and j in the embedding consists of at most two line segments. For element vertex i and set vertex j, we denote the line segment of edge (i, j) connected to i by $s_i(i, j)$ and the line segment connected to j by $s_j(i, j)$. If only one line segment builds the edge, then we split the line segment into two line segments of equal length and assign those to $s_i(i, j)$ respectively $s_j(i, j)$. By $l(\cdot)$ we refer to the Euclidean length of an edge respectively a line segment. Moreover, we let $f_i(i, j) = 1/l(s_i(i, j))$ and $f_j(i, j) = 1/l(s_j(i, j))$. The reason behind this definition is to define for any line segment, say of length l, a sequence of devices that make its cost a fixed constant, say 1. This can be achieved, roughly speaking, by placing l^2 evenly along the line segment, with each device bridging a gap of length $l/l^2 = 1/l$, at a total cost of $(1/l)^2 \times l^2 = 1$. In addition, we will modify the equidistant placement a little on one end of the device path, in order to make the connection work only in one direction.

We construct the instance $\hat{I}(T, \Delta n, r, Z)$ of Identical Device Placement from I as follows. We start with the description of the positions and transmission ranges of the devices in the transmission graph T. The device set V contains five different kinds of devices, called the *element*, the *set*, the *global destination*, the *element-set chain*, and the *set-destination chain devices*. We describe each kind separately (see Fig. 4.5 for an overview illustration).

4.5.1 Set Devices

For each set $j \in S = \{1\dots,|S|\}$, we place a set device v_j^s at position $\langle j_x, j_y\rangle$. These devices have a maximal transmission range equal to 2. We denote the union of all set devices by V^{s}.

4.5.2 Element Devices

For each element $i \in U = \{1,\dots,|U|\}$, we place an element device v_i^e at position $\langle i_x, i_y\rangle$. The maximal transmission range of such a device is set to $\max_{j:(i,j)\in E} 3/2 \cdot f_i(i,j)$. We denote the union of all element devices by V^{e}.

4.5.3 Global Destination Device

We place one single global destination device v^T at position $\langle 0, -M\rangle$, where M is a large number to be defined later. The maximal transmission range of this device is set arbitrarily (since it will not be used for transmission).

4.5.4 Element-Set Chain Devices

These devices model the connections between an element and each set to which the element belongs. For each element i and set S_j for which $i \in S_j$, we introduce an element-set chain $V^{i,j}$. The element-set chain $V^{i,j}$ looks as follows: we place devices $V^{i,j} = \left\{v_1^{i,j},\dots,v_{n_{i,j}}^{i,j}\right\}$, with $n_{i,j} = l(s_i(i,j))^2 + l(s_j(i,j))^2$, along the edges of the embedding (dotted line segments in the overview figure). The placement is done such that the distance between two consecutive devices on a line segment is at most $1/\hat{c}$, and the cost of a path from an element device to any of the at most three set devices is the same. To achieve this (see Fig. 4.6), we place a

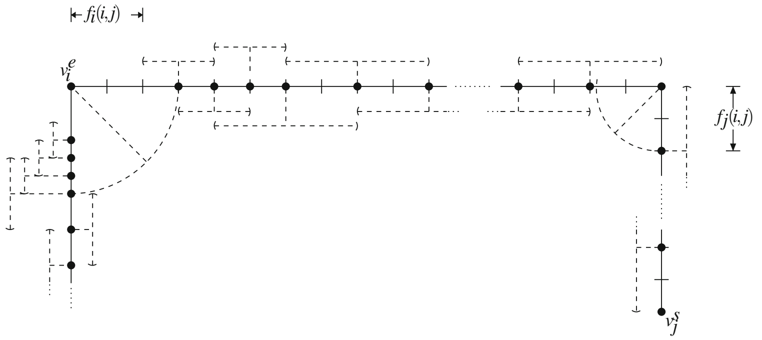


Fig. 4.6 Element-set chain $V^{i,j}$ between element device v_i^{e} and set device v_j^{s}. The dotted lines respectively circle segments indicate the maximal transmission range of a device

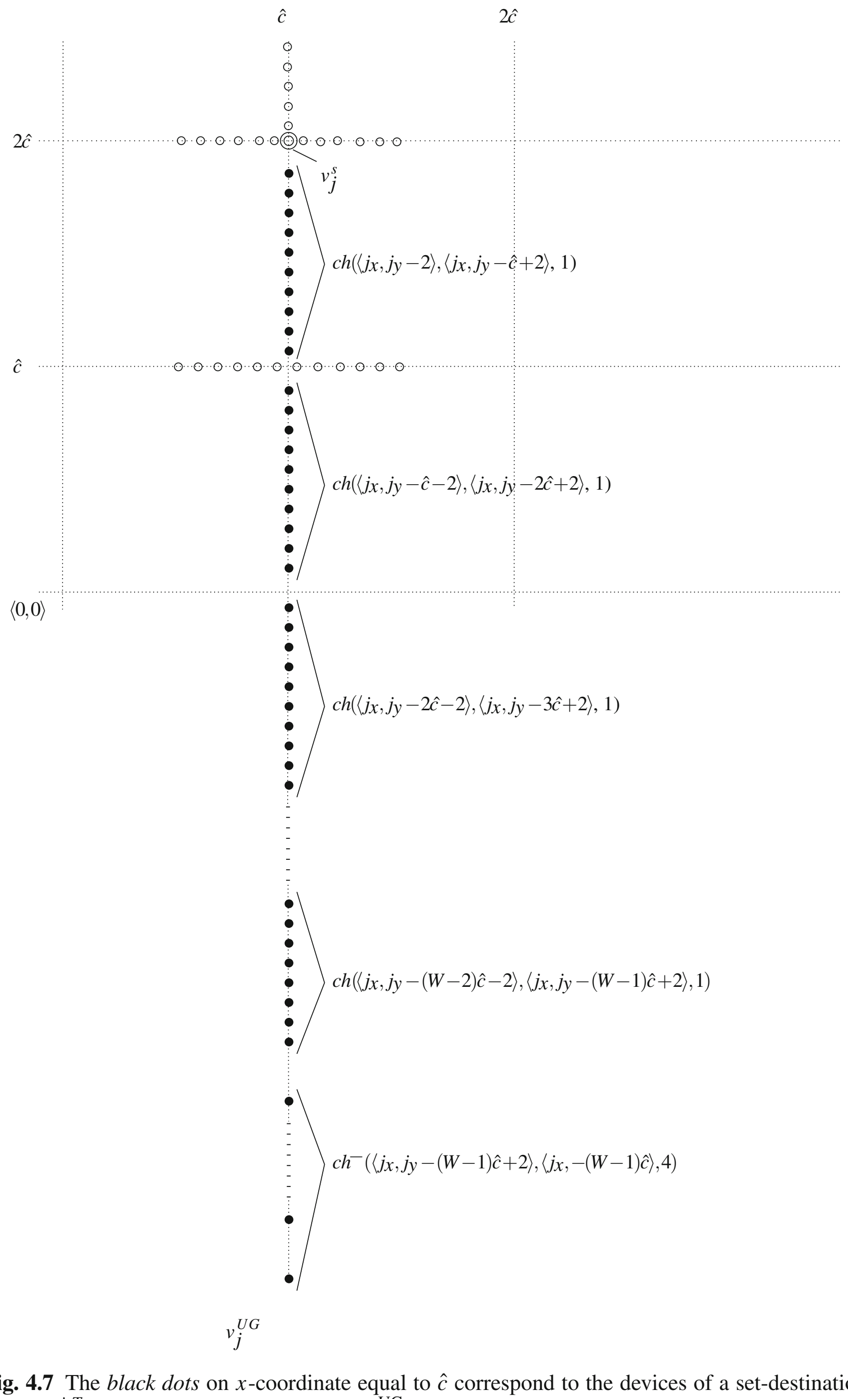
first device $v_1^{i,j}$ at distance $3/2 \cdot f_i(i,j)$ from the element device v_i^{e}. The maximal transmission range of this device is set to $1/2 \cdot f_i(i,j)$. Next, we place three devices $v_h^{i,j}$ for $h = 2, \ldots, 4$ at distance $3/2 \cdot f_i(i,j) + (h-1)/2 \cdot f_i(i,j)$ from the element device v_i^{e}. Devices $v_2^{i,j}$, $v_3^{i,j}$ have a maximal transmission range equal to $1/2 \cdot f_i(i,j)$, and $v_4^{i,j}$ one of $f_i(i,j)$. Devices $v_h^{i,j}$ for $h = 5, \ldots, l(s_i(i,j))^2$ with a maximal transmission range $f_i(i,j)$ are positioned at distances $(h-1) \cdot f_i(i,j)$ from the element device v_i^{e}. At the common endpoint of the two line segments $s_i(i,j)$ and $s_j(i,j)$, a further device $v_{l(s_i(i,j))^2+1}^{i,j}$ with a maximal transmission range $f_j(i,j)$ is placed. We continue with devices on line segment $s_j(i,j)$. We uniformly distribute devices on this line segment by placing devices $v_{l(s_i(i,j))^2+1+h}^{i,j}$ at distances $h \cdot f_j(i,j)$ from the common endpoint of $s_i(i,j)$ and $s_j(i,j)$, for $h = 1, \ldots, l(s_j(i,j))^2 - 1$. All devices on $s_j(i,j)$ have a maximal transmission range of $f_j(i,j)$. By the above construction, there is a path from any element device to a set device through the devices $\left(v_1^{i,j}, \ldots, v_{n_{i,j}}^{i,j}\right)$ with cost 2. Moreover, a path in the reverse direction is not possible due to the chosen maximal transmission ranges.

4.5.5 Set-Destination Chain Devices

These devices connect each set device to the global destination device. The (unavoidable) crossings with the element-set chains have to be constructed carefully for the reduction to work. In particular, by defining the transmission ranges of devices on the element-set chain to be small enough, we will make sure that at such a crossing, an element-set chain can never "switch" and continue on the set-destination chain. Moreover, we will make sure that no two set-destination chains cross. There is a set-destination chain $V^{j,T}$ from each set device v_j^{s} to the global destination device v^T, for $j \in \{1, \ldots, |S|\}$ (dashed lines in the overview figure). In detail, the following devices are set. Let $W = \lceil \frac{n}{2} \rceil$. We distinguish three cases depending on the free direction of the set vertex j. Note that we can choose the graph drawing from [17] such that for no set vertex the free direction is upward.

Free direction of set vertex j is downward. We place chains of devices interrupted by longer distances without devices in the free direction (see Fig. 4.7 for a schematic illustration). More precisely, we place $(W-1)$ chains $\mathrm{ch}(\langle j_x, j_y - h \cdot \hat{c} - 2\rangle, \langle j_x, j_y - (h+1)\hat{c} + 2\rangle, 1)$, where $h = 0, \ldots, W-2$. The devices have a maximal transmission range of $1/(\hat{c}-4)$ except the last device on each chain, which has a maximal transmission range of 4. Finally, we place one more chain $\mathrm{ch}^{-}(\langle j_x, j_y - (W-1)\hat{c} + 2\rangle, \langle j_x, -(W-1)\hat{c}\rangle, 4)$. We denote the "last" device, at position $\langle j_x, -(W-1)\hat{c}\rangle$, by v_j^{UG}. The maximal transmission range of these devices except v_j^{UG} is set such that any device can reach the next device on the chain, and v_j^{UG}'s maximal transmission range equals d_{gap}, which will be defined later.

Free direction of set vertex j is to the left. Again, we place chains of devices interrupted by longer distances without devices in the free direction. More precisely,



$\hat{c}$
$2\hat{c}$
$2\hat{c}$
$\hat{c}$
$\langle 0,0\rangle$
v_j^s
$ch(\langle jx, jy-2\rangle, \langle jx, jy-\hat{c}+2\rangle, 1)$
$ch(\langle jx, jy-\hat{c}-2\rangle, \langle jx, jy-2\hat{c}+2\rangle, 1)$
$ch(\langle jx, jy-2\hat{c}-2\rangle, \langle jx, jy-3\hat{c}+2\rangle, 1)$
$ch(\langle jx, jy-(W-2)\hat{c}-2\rangle, \langle jx, jy-(W-1)\hat{c}+2\rangle, 1)$
$ch^{-}(\langle jx, jy-(W-1)\hat{c}+2\rangle, \langle jx, -(W-1)\hat{c}\rangle, 4)$
v_j^{UG}

Fig. 4.7 The *black dots* on x-coordinate equal to $\hat{c}$ correspond to the devices of a set-destination chain $V^{j,T}$ from set device v_j^{s} to device v_j^{UG}. The *circles* are devices belonging to element-set chains

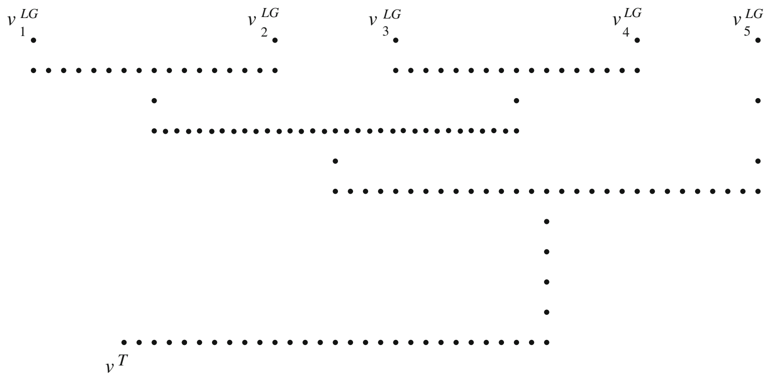


Fig. 4.8 Second part of set-destination chains $V^{1,T}$ to $V^{5,T}$ from devices v_j^{LG} to the global destination device v^T

we place $(W-1)$ chains $\mathrm{ch}(\langle j_x - h\cdot\hat{c} - 2, j_y\rangle, \langle j_x - (h+1)\hat{c} + 2, j_y\rangle, 1)$ for $h = 0, \ldots, W-2$. The devices in the chains have a maximal transmission range of $1/(\hat{c}-4)$, except the last device on each chain, having a maximal transmission range of 4. We make sure that no two such chains intersect by moving further outward from the bounding box. To this end, we add a left going chain $\mathrm{ch}^{-}(\langle j_x - (W-1)\hat{c}+2, j_y\rangle, \langle -(W-1)\hat{c} - j_y, j_y\rangle, 2)$, followed by a downward chain $\mathrm{ch}^{-}(\langle -(W-1)\hat{c} - j_y, j_y\rangle, \langle -(W-1)\hat{c} - j_y, -(W-1)\hat{c}\rangle, 2)$. We denote the device at position $\langle -(W-1)\hat{c} - j_y, -(W-1)\hat{c}\rangle$ by v_j^{UG}. The maximal transmission range of the devices in the latter two chains except v_j^{UG} is set such that any device can reach the next device on the chain, and v_j^{UG}'s maximal transmission range equals d_{gap}.

Free direction of set vertex j is to the right. This case is similar to the previous case except that the chain first moves in the other direction. More precisely, we place $(W-1)$ chains $\mathrm{ch}(\langle j_x + h\cdot\hat{c} + 2, j_y\rangle, \langle j_x + (h+1)\hat{c} - 2, j_y\rangle, 1)$ for $h = 0, \ldots, W-2$. The devices have a maximal transmission range of $1/(\hat{c}-4)$, except the last device on each chain, having a maximal transmission range of 4. Further, we add a chain $\mathrm{ch}^{-}(\langle j_x + (W-1)\hat{c}+2, j_y\rangle, \langle \hat{c}W + (W-1)\hat{c} + j_y, j_y\rangle, 2)$, and a chain $\mathrm{ch}^{-}(\langle \hat{c}W + (W-1)\hat{c} + j_y, j_y\rangle, \langle \hat{c}W + (W-1)\hat{c} + j_y, -(W-1)\hat{c}\rangle, 2)$. We denote the device at position $\langle \hat{c}W + (W-1)\hat{c} + j_y, -(W-1)\hat{c}\rangle$ by v_j^{UG}. The maximal transmission range of the devices in the latter two chains except v_j^{UG} is set such that any device can reach the next device on the chain, and v_j^{UG}'s maximal transmission range equals d_{gap}.

The construction of the set-destination chains yields a device v_j^{UG} for every set device v_j^{s} on the horizontal line $y = -(W-1)\hat{c}$. Moreover, there exists a path between any set device v_j^{s} and the corresponding device v_j^{UG} with cost $2^2 + (W-1)\cdot 1 + (W-2)\cdot 4^2 + 4 = 17W - 25$.

As a next step in the construction of the set-destination chains, we build a set of devices resembling a tree (see Fig. 4.8 for a schematic illustration). We place devices on the $h = \lceil \log |S| \rceil$ consecutive integer y-coordinates $y_\ell = -(W-1)\hat{c} - d_{\text{gap}} - \ell$, for $\ell = 0, \ldots, h$. We refer to these y-coordinates as levels, where level 0 is the level with the largest y-coordinate. The number of devices placed on level ℓ equals $n_\ell = \lceil |S|/2^\ell \rceil$. On level 0, the n_0 devices have the same x-coordinate as the devices v_j^{UG}. We call these devices v_j^{LG} and refer to the gap between v_j^{UG} and v_j^{LG} as the *large gap*. On level 1, we compute the x-coordinates from two devices of level 0. We start at the left side. We place a device on level 1, with its x-coordinate halfway between the two devices with smallest x-coordinates from level 0. We repeat this procedure with the two devices with next largest x-coordinate from level 0 until no two devices are left. If there is one device left, then we place a device with same x-coordinate on level 1. We iteratively continue this procedure on the other levels until $\ell = h$. The maximal transmission range for all these devices is set to $1/2$, except for the rightmost device on a level with an odd number of devices, which has maximal transmission range 1. Between the levels, we add further devices to connect devices u, v that have been paired up: we add a chain $\text{ch}(\langle u_x, u_y - 1/2\rangle, \langle (u_x + v_x)/2, u_y - 1/2\rangle, 1/2)$ and a chain $\text{ch}^-(\langle v_x, v_y - 1/2\rangle, \langle (u_x + v_x)/2, v_y - 1/2\rangle, 1/2)$ with maximal transmission ranges set such that any device can reach the next device on its chain. The device at position $\langle (u_x + v_x)/2, u_y - 1/2\rangle$ has a maximal transmission range of $1/2$. To complete the set-destination chains, two chains of devices are placed between the single device on level h and the global destination device v^T with a path of cost 2, one chain on the vertical line segment down to y-coordinate equal to $-M$ and one chain on the horizontal line toward v^T, with maximal transmission ranges sufficiently large to reach the next device toward the global destination device.

The construction of the set-destination chains leads to a path between any element device and the global destination device v^T with total cost of $17W - 21 + (d_{\text{gap}})^2 + \lceil \log S \rceil$. Depending on whether the element is a member of two or three sets there are two or three paths with this cost per element device. Since $\hat{c}$ is polynomial in n, the total number of devices in the instance $\hat{I}$ is polynomial in the input parameters.

As a next step in the reduction, we describe the commodity set K of the transmission graph. We construct $k = |U|$ commodities with each element device $v_i^{\text{e}}, i = 1, \ldots, |U|$, being a source and device v^T being the global destination device for each such source. Without additional devices, a shortest path for any commodity goes from the source over an element-set chain to a set device to which the source belongs and from there along a set-destination chain to the global destination device. Note that we constructed the crossings of element-set chains with set-destination chains in such a way that the transmission ranges of element-set chain devices are not large enough to switch to the set-destination chain. Furthermore, even though switching from a set-destination chain to an element-set chain is possible at such a crossing, it will only make the path longer, by more than 1 cost unit.

The maximal transmission range r for the additional devices is set to a positive value t that is small enough to make sure that unless each device is placed in a gap that a set-destination chain uses, it will not contribute sufficiently to cost savings, say $t < \frac{1}{2bd_{\text{gap}}^2}$. Finally, we set $\Delta n = b$ and $Z = |U| \cdot 2t(d_{\text{gap}} - t)$.

This justifies to set the scaling factor $\hat{c}$ to $4n^3$. Furthermore, we set the distance of the gap d_{gap} to n^2, and the y-coordinate $-M$ of the global destination device to $-n^4$. This completes the construction, which is computable in polynomial time.

To complete the reduction we show that I is a YES-instance of ++X3C-3 if and only if $\hat{I}$ is a YES-instance of Identical Device Placement. We first show the forward direction, that is, given a cover of size b we can construct a device placement with an improvement Z. Let the indices of the b sets in the ++X3C-3 solution be $(\text{sol}_1, \ldots, \text{sol}_b)$. Then, consider the placement of Δn additional devices at positions $\left\langle \left(v^{\text{UG}}_{\text{sol}_i}\right)_x, \left(v^{\text{LG}}_{\text{sol}_i}\right)_y + t \right\rangle$ for $i = 1, \ldots, \Delta n$, i.e., as far above as possible from the lower ends of the large gaps on the set-destination chains so that the transmission range t still suffices to reach the next device. For each commodity j, the cost between v^{UG}_j and v^{LG}_j goes from d^2_{gap} down to $d_{\text{gap}} - t^2 + t^2$, a savings of $2t(d_{\text{gap}} - t)$ per commodity, resulting in a total improvement $Z = |U| \cdot 2t(d_{\text{gap}} - t)$.

Now we show that a placement of Δn additional devices that saves a cost of at least $Z = |U| \cdot 2t(d_{\text{gap}} - t)$ implies a solution to the ++X3C-3 instance.

Observe that even after an arbitrary placement of all additional devices, it still does not pay for a set-destination path to switch to an element-set path. The reason is that the highest possible savings that arise from placing all additional devices are still less than the extra cost from switching. To see this, recall that the extra switching cost is more than 1, and note that the highest savings can be achieved when additional devices bridge gaps that are as large as possible. The largest useful gaps in the entire construction are our large gaps, of size d_{gap}; all larger ones are useless, because bridging them with additional devices still leaves them too large to be part of any communication path. The cost savings for a large gap, however, are smaller than 1 even if all additional devices are placed optimally in a single large gap. To see this, note that an optimal placement of b additional devices into a large gap puts them in series at distances t from each other and from the low vertex $v^{\text{LG}}_{\text{sol}_i}$ of the gap, with a cost of $(d_{\text{gap}} - bt)^2 + bt^2$ instead of d^2_{gap}, a savings of less than 1 for our chosen value of $t < \frac{1}{2bd^2_{\text{gap}}}$. Since the remaining part of the large gap after placing all b devices has size $d_{gap} - bt$, this is still the largest useful gap in the construction, and therefore any (other) combination of placing the b devices into gaps also leads to savings smaller than 1. Hence, no set-destination path switches at a crossing.

Furthermore, observe that the transmission ranges of the new devices are too small to let any element-set path switch at a crossing. To see this, recall that any set-destination path has a gap of size 4 at a crossing, and therefore from a device on an element-set chain a gap of size almost 2 must be bridged, but all new devices together can only bridge a gap of size bt, which is less than $\frac{1}{2d_{\text{gap}}}$ and therefore less than $\frac{1}{2n^2}$, not enough by a large margin.

That is, all element-set paths and set-destination paths must use the chains prescribed in our construction and must go over a large gap (possibly shortened by extra devices) to the destination. It therefore remains to show that the only way to achieve a savings of $|U| \cdot 2t(d_{\text{gap}} - t)$ is to place one device each into the gaps that describe the solution (i.e., the chosen subsets) of the instance of ++X3C-3. Because the highest possible savings per new device are $2t(d_{\text{gap}} - t)$, and this can only be achieved by placing only single devices in large gaps, the required savings of $|U| \cdot 2t(d_{\text{gap}} - t)$ can only be achieved by letting each element-destination path go over a large gap with a single new device at distance t from its bottom vertex. But if this is possible, then the gaps that contain the new devices describe a solution to the ++X3C-3 instance.

Acknowledgments We acknowledge the support of this work by the Swiss National Science Foundation through the NCCR project Mobile Information and Communication Systems (MICS). Parts of this work appeared in preliminary form [4] at the third international Algosensors workshop.

References

1. P. Agarwal, O. Schwarzkopf, and M. Sharir. The overlay of lower envelopes and its applications. *Discrete & Computational Geometry*, 15(1):1–13, Springer, New York, NY, 1996.
2. S. Albers, S. Eilts, E. Even-Dar, Y. Mansour, and L. Roditty. On Nash equilibria for a network creation game. In: *Proceedings SODA 2006*, pages 89–98, Miami, Florida, 2006.
3. L. Anderegg and S. Eidenbenz. Ad hoc-VCG: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In: *Proceedings MOBICOM 2003*, pages 245–259, San Diego, California, 2003.
4. L. Anderegg, S. Eidenbenz, L. Peeters, and P. Widmayer. Optimal placement of ad-hoc devices under a vcg-style routing protocol. In M. Kutylowski, J. Cichon, and P. Kubiak, editors, *ALGOSENSORS*, vol. 4837 of *Lecture Notes in Computer Science*, pages 58–70. Springer, New York, NY, 2007.
5. E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. In: *Proceedings FOCS 2004*, pages 295–304, Rome, Italy, 2004.
6. E. Anshelevich, A. Dasgupta, E. Tardos, and T. Wexler. Near-optimal network design with selfish agents. In: *Proceedings STOC 2003*, pages 511–520, 2003.
7. J. Corbo and D. Parkes. The price of selfish behavior in bilateral network formation. In: *Proceedings PODC 2005*, pages 99–107, Las Vegas, Nevada, 2005.
8. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, New York, NY, 1997.
9. M. Dyer and A. Frieze. Planar 3DM is np-complete. *Journal of Algorithms*, 7:174–184, 1986.
10. S. Eidenbenz, G. Resta, and P. Santi. The commit protocol for truthful and cost-efficient routing in ad hoc networks with selfish nodes. *IEEE Transactions on Mobile Computing*, 7(1):19–33, 2008.
11. A. Fabrikant, A. Luthra, E. Maneva, C. Papadimitriou, and S. Shenker. On a network creation game. In: *Proceedings PODC 2003*, Boston, Massachusetts, 2003.
12. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.
13. O. Goussevskaia, M. Halldórsson, R. Wattenhofer, and E. Welzl. Capacity of Arbitrary Wireless Networks. In: *28th Annual IEEE Conference on Computer Communications (INFOCOM)*, Rio de Janeiro, Brazil, April, 2009.
14. O. Goussevskaia, Y. A. Oswald, and R. Wattenhofer. Complexity in Geometric SINR. In: *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, Montreal, Canada, September, 2007.
15. A. Hayrapetyan, E. Tardos, and T. Wexler. A network pricing game for selfish traffic. In: *Proceedings PODC 2005*, pages 284–291, Las Vegas, Nevada, 2005.

16. M. Hoefer and P. Krysta. Geometric Network Design with Selfish Agents. In: *Proceedings of 11th Annual International Conference on Computing and Combinatorics (COCOON 2005)*, pages 167–178, Kunming, Yunnan, 2005.
17. G. Kant. Drawing planar graphs using the lmc-ordering. In: *Proceedings FOCS 1992*, pages 101–110, Pittsburgh, Pennsylvania, 1992.
18. S. Krumke. *The Approximability of Location and Network Design Problems*. PhD thesis, University of Wuerzburg, 1996.
19. A. Mas-Colell, M. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, 1995.
20. C. Thielen and S. O. Krumke. Truthful mechanisms for selfish routing and two-parameter agents. In: *SAGT*, pages 36–47, Paphos, Cyprus, 2009.

Chapter 5
Population Protocols and Related Models

Paul G. Spirakis

Abstract This is a joint work with Ioannis Chatzigiannakis and Othon Michail. We discuss here the population protocol model and most of its well-known extensions. The population protocol model aims to represent sensor networks consisting of tiny computational devices with sensing capabilities that follow some unpredictable and uncontrollable mobility pattern. It adopts a minimalistic approach and, thus, naturally computes a quite restricted class of predicates and exhibits almost no fault tolerance. Most recent approaches make extra realistic and implementable assumptions, in order to gain more computational power and/or speedup the time to convergence and/or improve fault tolerance. In particular, the mediated population protocol model, the community protocol model, and the PALOMA model, which are all extensions of the population protocol model, are thoroughly discussed. Finally, the inherent difficulty of verifying the correctness of population protocols that run on complete communication graphs is revealed, but a promising algorithmic solution is presented.

5.1 Introduction

Wireless Sensor Networks (WSNs) will play an increasingly important role in critical systems' infrastructure and should be correct, reliable, and robust. Formal specification helps to obtain not only a better (more modular) description, but also a clear understanding and an abstract view of a system [8]. Given the increasing sophistication of WSN algorithms and the difficulty of modifying an algorithm once the network is deployed, there is a clear need to use formal methods to validate system performance and functionality prior to implementing such algorithms [34]. Formalanalysis requires the use of models, trusted to behave like a real system. It

P.G. Spirakis (✉)
Research Academic Computer Technology Institute (RACTI) Patras, Greece and Department of Computer Engineering and Informatics, University of Patras,
e-mail: spirakis@cti.gr

This work has been partially supported by the ICT Programme of the European Union under contract number ICT-2008-215270 (FRONTS).

is therefore critical to find the correct abstraction layer for the models and to verify the models.

Toward providing a concrete and realistic model for future sensor networks, Angluin et al. [2] introduced the notion of a computation by a population protocol. Due to the minimalistic nature of their model, individual agents are extremely limited and can be represented as finite-state machines. The computation is carried out by a collection of agents, each of which receives a piece of the input. Information can be exchanged between two agents whenever they come into contact with (or sufficiently close to) each other. The goal is to ensure that every agent can eventually output the value that is to be computed. The critical assumptions that diversify the population protocol model from traditional distributed systems are that the interaction pattern is inherently nondeterministic and that the protocols' description is independent of the population size (that is, need $\mathcal{O}(1)$ total memory capacity in each agent). The latter is known as the *uniformity property* of population protocols. Moreover, population protocols are *anonymous* since there is no room in the state of an agent to store a unique identifier.

The population protocol model was designed to represent sensor networks consisting of very limited mobile agents with no control over their own movement. It also bears a strong resemblance to models of interacting molecules in theoretical chemistry [26, 27]. The defining features of the population protocol model are as follows:

1. Anonymous, finite-state agents. The system consists of a large population of indistinguishable finite-state agents.
2. Computation by direct interaction. In the original model, agents do not send messages or share memory; instead, an interaction between two agents updates both of their states according to a global transition table. The actual mechanism of such interactions is abstracted away.
3. Unpredictable interaction patterns. The choice of which agents interact is made by an adversary. Agents have little control over which other agents they interact with, although the adversary may be limited to pairing only agents that are adjacent in an interaction graph, typically representing distance constraints or obstacle presence. A strong global fairness condition is imposed on the adversary to ensure that the protocol makes progress (e.g., the adversary cannot keep the agents forever disconnected).
4. Distributed inputs and outputs. The input to a population protocol is distributed across the agents of the entire population. In what concerns predicates, all agents are expected to give the correct output value (which is known as the *predicate output convention* [2]); thus, the output is collected from any agent in the population (after, of course, the computation has stabilized).
5. Convergence rather than termination. Population protocols generally cannot detect when they have finished; instead, the agents' outputs are required to converge after some finite time to a common correct value.

The population protocol model was inspired in part by work by Diamadi and Fischer [23] on trust propagation in a social network. The *urn automata* of [3] can be seen as a first draft of the model that retained in vestigial form several features of classical automata: instead of interacting with each other, agents could only interact with a finite-state controller, complete with input tape. The motivation given for the current model in [2] was the study of sensor networks in which passive agents were carried along by other entities; the canonical example was sensors attached to a flock of birds. The name of the model was chosen by analogy to population processes [33] in probability theory.

The initial goal of the model was to study the *computational limitations* of cooperative systems consisting of many limited devices (agents), imposed to passive (but *fair*) communication by some *scheduler*. Much work showed that there exists an exact characterization of the computable predicates: they are precisely the *semilinear predicates* or equivalently the predicates definable by first-order logical formulas in *Presburger arithmetic* [2, 5–7]. More recent work has concentrated on performance, supported by a random scheduling assumption. Chatzigiannakis et al. [16] proposed a collection of *fair* schedulers and examined the performance of various protocols. Chatzigiannakis et al. [12] went one step further by proposing a generic definition of probabilistic schedulers and proving that the schedulers of [16] are all fair with probability 1 and revealed the need for the protocols to adapt when natural modifications of the mobility pattern occur. Bournez et al. [11] and Chatzigiannakis and Spirakis [19] considered a huge population hypothesis (population going to infinity) and studied the dynamics, stability, and computational power of probabilistic population protocols by exploiting the tools of continuous nonlinear dynamics. In [11] it was also proven that there is a strong relation between classical finite population protocols and models given by ordinary differential equations.

There exist a few extensions of the population protocol model in the relevant literature to more accurately reflect the requirements of practical systems. In [1] they studied what properties of restricted communication graphs are stably computable, gave protocols for some of them, and proposed the model extension with *stabilizing inputs*. The results of [5] show that again the semilinear predicates are all that can be computed by this model. Finally, some works incorporated agent failures [22] and gave to some agents slightly increased computational power [9] (heterogeneous systems). For an excellent introduction to most of the preceding subjects see [7].

In this chapter we start by presenting in detail the basic population protocol model. Unfortunately, the class of solvable problems by this theoretical model is fairly small. For instance, it does not include multiplication. Moreover, even for this restricted class, algorithms tolerate no failures or, at worst, a fixed number of benign failures [22]. Therefore, we present four interesting extensions of the population protocol model that investigate the computational benefits of cooperative systems when adding new features (e.g., to the hardware of the devices). The extended models are summarized as follows:

- First, based on [17] (see also [18]), the population protocol model is extended to include a *Mediator*, i.e., a global storage capable of storing very limited information for each communication arc (the state of the arc). When pairs of agents interact, they can read and update the state of the link (arc). The extended model is called the *Mediated Population Protocol* (*MPP*) model. Interestingly, although anonymity and uniformity are preserved in this model, *the presence of a mediator provides us with significantly more computational power* and gives birth to a new collection of interesting problems in the area of tiny networked and possibly moving artifacts; based on this model we can build systems with the ability of computing subgraphs and solve optimization problems concerning the communication graph. Moreover, as we shall see, MPPs are capable of computing nonsemilinear predicates and here any stably computable predicate belongs to *NSPACE(m)*, where m denotes the number of edges of the communication graph.

- One of the most interesting and applicable capabilities of the mediated population protocol model is its ability to decide graph properties. To understand the properties of the communication graph is an important step in almost any distributed system. In particular, if we temporarily disregard the input notion of the population and assume that all agents simply start from a unique initial state (and the same holds for the edges), then we obtain another interesting model that is called the *GDM* (standing for Graph Decision Mediated) model [15]. When GDM protocols are executed fairly on any communication graph G, after a finite number of steps stabilize to a configuration where all agents give 1 as output if G belongs to a graph language L, and 0 otherwise. This is motivated by the idea of having protocols that eventually accept all communication graphs (on which they run) that satisfy a specific property, and eventually reject all remaining communication graphs. The motivation for studying a simplified version of the mediated population protocol model is that it enables us to study what graph properties are stably computable by the mediated model without the need to keep in mind its remaining parameters (which, as a matter of fact, are a lot).
- Another direction for extending the population protocol model is to assume the existence of a unique identifier for each agent. This is a natural extension since, although a tiny device's memory is often very constrained, it is usually sufficient to store a unique identity. In fact, in most modern tiny devices, the communication module is often equipped with a unique identifier. For example, they might contain Maxim's DS2411 chip, which stores just 64 bits of ROM and is set by the factory to store a unique serial number. This idea gave birth to the *Community Protocol* model [29]. In this model, all n agents have unique identifiers (ids) and can store $\mathcal{O}(1)$ other agents' ids. The ids are stored in ROM (as in the DS2411 chip), so that Byzantine agents cannot alter their ids. The usage of ids is restricted to their fundamental purpose, *identification*, by assuming that algorithms can only compare ids (an algorithm cannot, for example, perform arithmetic on ids). In addition to *having* ids, the ability of agents to *remember* other ids is crucial as, otherwise, the model would be as weak as population protocols. The computational power of this extension is greatly increased; a community protocol of n agents can simulate a nondeterministic Turing Machine of $\mathcal{O}(n \log n)$ space. In particular, it can compute any symmetric predicate in $NSPACE(n \log n)$. Moreover, as in the population protocol model, a single algorithm must work for all values of n. Furthermore, the simulation is resilient to a constant number of Byzantine failures. So, although community protocols only make a rational

additional assumption (that is, the ids equipment), they are much more powerful than population protocols: they solve a much wider class of problems and tolerate Byzantine failures.

- Finally, we present another extension called the *PALOMA* model [14]. In this model, the system consists of PAssively mobile LOgarithmic space MAchines. The idea is to provide each agent with a memory whose size is logarithmic in the population size, which seems a very natural assumption: only 266 bits are required for 2^{266} agents (which is an astronomical population size)! Moreover, we can think of an agent as a small Turing Machine, which also seems natural: mobile phones, PDAs and many other common mobile devices are in fact sophisticated Turing Machines. The PALOMA model is also extremely strong, since it can stably compute any symmetric predicate in $NSPACE\ (n \log n)$.

A very important aspect of WSNs is to provide solutions that are verifiably correct, in the sense of giving a "proof" that the solution will work, given the application goals and network set-up. Population protocol models can detect errors in the design that are not so easily found using emulation or testing. Formal analysis techniques are also supported by (semi-)automated tools. Such tools can also detect errors in the design and they can be used to establish correctness. Model checking is an exhaustive state space exploration technique that is used to validate formally specified system requirements with respect to a formal system description [21]. Such a system is verified for a fixed configuration; so in most cases, no general system correctness can be obtained. Using some high-level formal modeling language, automatically an underlying state space can be derived, be it implicitly or symbolically. The system requirements are specified using some logical language, like LTL, CTL or extensions thereof [32]. Well-known and widely applied model checking tools are SPIN [31], Uppaal [10] (for timed systems), and PRISM [30] (for probabilistic systems). The system specification language can, e.g., be based on process algebra, automata or Petri nets. However, model checking suffers from the so-called state explosion problem, meaning that the state space of a specified system grows exponentially with respect to its number of components. The main challenge for model checking lies in modeling large-scale dynamic systems.

The important feature that diversifies the population protocol model from traditional distributed systems is that the protocol specifications are independent of the population size which makes them suitable for the verification of protocols that target systems spanning thousands of objects. Evaluating if a property is valid or not in the system can be done with a number of components that is independent to the size of the population. The most important factor to decide the reachability of a certain configuration is the size of the protocol. Toward further minimizing the configuration space of the protocols we can apply the protocol composition methodology. This approach states that one may reduce a protocol into two (or more) protocols of reduced state space that maintain the same correctness and efficiency properties. The combination of the above helps overcome the state explosion problem and speed up the verification process. We expect that population protocol models will be used to model such networks and the interactions, as dictated by the MAC protocol or the overall protocol stack, providing the ability, in a formal and modern way, to define the system in a minimalist way (in contrast to other approaches).

Section 5.2 discusses the population protocol model of Angluin et al. [2]. Section 5.3 deals with a first extension of the population protocol model, the mediated population protocol model [17]. Section 5.4 goes one step further in the investigation of the mediated population protocol model by focusing on its ability to decide interesting graph properties. The simplified version of the mediated population protocol model discussed there is the GDM model [15]. In Sect. 5.5, the community protocol model of Guerraoui and Ruppert [29] is discussed and in Sect. 5.6 the PALOMA model [14]. Both models have the same computational power and are particularly powerful. Section 5.7 deals with correctness of population protocols that run on complete communication graphs. In particular, it focuses on the problem of algorithmically verifying whether a given population protocol is correct w.r.t. its specifications and is based on [13]. The problem is shown to be hard, but a promising algorithmic solution is presented. Finally, Sect. 18.6 discusses some interesting open problems in the area of small passively mobile communicating devices.

For a good introduction and definitions on Computational Complexity (see e.g., [35]).

5.2 Population Protocols

We begin with a formal definition of the population protocol model proposed in a seminal work of Angluin et al. [2]. The model represents sensor networks consisting of extremely limited agents that may move and interact in pairs.

5.2.1 The Model

Definition 1 A *population protocol* (PP) is a 6-tuple (X, Y, Q, I, O, δ), where X, Y, and Q are all finite sets and

1. X is the *input alphabet*,
2. Y is the *output alphabet*,
3. Q is the set of *states*,
4. $I : X \to Q$ is the *input function*,
5. $O : Q \to Y$ is the *output function*, and
6. $\delta : Q \times Q \to Q \times Q$ is the *transition function*.

If $\delta(a, b) = (a', b')$, we call $(a, b) \to (a', b')$ a *transition* and we define $\delta_1(a, b) = a'$ and $\delta_2(a, b) = b'$. We call δ_1 the *initiator's acquisition* and δ_2 the *responder's acquisition*.

A population protocol $\mathcal{A} = (X, Y, Q, I, O, \delta)$ runs on a communication graph $G = (V, E)$ with no self-loops and no multiple edges. From now on, we will denote by n the number of nodes of the communication graph and by m the number of its edges. Initially, all agents (i.e. the elements of V) receive a global start signal, sense their environment and each one receives an input symbol from X. All agents are initially in a special empty state $\sqcup \notin Q$. When an agent receives an input symbol σ, applies the input function to it and goes to its initial state $I(\sigma) \in Q$. An adversary scheduler selects in each step a directed pair of distinct agents $(u, \upsilon) \in E$ (that is, $u, \upsilon \in V$ and $u \neq \upsilon$) to interact. The interaction is established only if both agents are not in the empty state (they must both have been initialized). Assume that the scheduler selects the pair (u, υ), that the current states of u and υ are $a, b \in Q$, respectively, and that $\delta(a, b) = (a', b')$. Agent u plays the role of the *initiator* in the interaction (u, υ) and υ that of the *responder*. During their interaction u and υ apply the transition function to their directed pair of states (to be more precise, the initiator applies δ_1 while the responder δ_2) and, as a result, u goes to a' and υ to b' (both update their states according to δ).

A *configuration* is a snapshot of the population states. Formally, a configuration is a mapping $C : V \to Q$ specifying the state of each agent in the population. C_0 is the initial configuration (for simplicity, we assume that all agents apply the input function at the same time, which is one step before C_0, so in C_0 all empty states have been already replaced, and that is the reason why we have chosen not to include $\sqcup$ in the model definition) and, for all $u \in V$, $C_0(u) = I(x(u))$, where $x(u)$ is the input symbol sensed by agent u. Let C and C' be configurations, and let u, υ be distinct agents. We say that C goes to C' via encounter $e = (u, \upsilon)$, denoted $C \overset{e}{\to} C'$, if

$$\begin{aligned} C'(u) &= \delta_1(C(u), C(\upsilon)), \\ C'(\upsilon) &= \delta_2(C(u), C(\upsilon)), \text{ and} \\ C'(w) &= C(w) \text{ for all } w \in V - \{u, \upsilon\}, \end{aligned}$$

that is, C' is the result of the interaction of the pair (u, υ) under configuration C and is the same as C except for the fact that the states of u, υ have been updated according to δ_1 and δ_2, respectively. We say that C can *go to C' in one step*, denoted $C \to C'$, if $C \overset{e}{\to} C'$ for some encounter $e \in E$. We write $C \overset{*}{\to} C'$ if there is a sequence of configurations $C = C_0, C_1, \ldots, C_t = C'$, such that $C_i \to C_{i+1}$ for all i, $0 \leq i < t$, in which case we say that C' is *reachable* from C.

An *execution* is a finite or infinite sequence of configurations $C_0, C_1, C_2, \ldots$, where C_0 is an initial configuration and $C_i \to C_{i+1}$, for all $i \geq 0$. We have both finite and infinite kinds of executions since the scheduler may stop in a finite number of steps or continue selecting pairs for ever. Moreover, note that, according to the preceding definitions, a scheduler may partition the agents into noncommunicating clusters. If that is the case, then it is easy to see that no meaningful computation is possible. To avoid this unpleasant scenario, a strong global *fairness condition* is imposed on the scheduler to ensure that the protocol makes progress. Formally, an infinite execution is *fair* if for every pair of configurations C and C' such that $C \to C'$, if C occurs infinitely often in the execution, then C' also occurs infinitely often in the execution. A scheduler is fair if it always leads to fair executions. A *computation* is an infinite fair execution.

The above fairness condition, although at first sight may seem too strong, is in fact absolutely natural. The reason is that in most natural systems, between those under consideration, the passive mobility pattern that the agents follow will be the result of some natural phenomenon, like, for example, birds flying, river flow, and so on, that usually follows some probability distribution or, possibly, a collection of such probability distributions. Most of these schedulers, as indicated by [12], satisfy the above fairness condition; they only have to also satisfy some natural properties.

The following are two critical properties of population protocols:

1. *Uniformity*: Population protocols are uniform. This means that any protocol's description is independent of the population size. Since we assume that the agents have finite storage capacity, and independent of the population size, uniformity enables us to store the protocol code in each agent of the population.
2. *Anonymity*: Population protocols are anonymous. The set of states is finite and does not depend on the size of the population. This implies that there is no room in the state of an agent to store a unique identifier, and, thus, all agents are treated in the same way by the transition function.

Example 1 A very celebrated population protocol is the "flock of birds" (or "*count to five*") protocol. Every bird in a particular flock is equipped with a sensor node that can determine whether the bird's temperature is elevated or not, and we wish to know whether at least five birds in the flock have elevated temperatures. Moreover, we assume that all ordered pairs of sensor nodes are permitted interaction. This was the motivating scenario of population protocols [2].

We think as follows. The sensor senses the temperature of its corresponding bird (its carrier) and if it is found elevated it outputs 1, otherwise 0. As soon as the agent receives a global start signal (e.g. from a base station) it reads its sensor's output $\sigma \in \{0, 1\}$ and applies to it the input function I. We can assume here that I maps 0 to initial state q_0 and 1 to q_1. This means that the number of agents that are in state q_1 under the initial configuration is equal to the number of sick birds, while all remaining agents are in state q_0. Now, when two agents interact, the initiator sets its state index to the sum of the state indices and the responder goes to q_0, except for the case in which the sum of the indices is at least 5. In the latter case both agents set their indices to 5. The idea is to try aggregating the initial number of 1 index to one agent's state index. Note that the sum of nonzero indices is always equal to the number of sick birds; obviously, this holds until index 5 first appears. But what about the output of the protocol? If an agent gets q_5 then it knows that initially at least five birds were sick, and it outputs the value 1 in order to indicate this fact, and eventually q_5 is propagated to all agents. Otherwise, it outputs 0 because it may still have partial information.

Let us now formalize the above description. The "*flock of birds*" protocol is $\mathcal{F} = (X, Y, Q, I, O, \delta)$. The input and output alphabets are $X = Y = \{0, 1\}$, the set of states is $Q = \{q_0, q_1, \ldots, q_5\}$, the input function I maps 0 to q_0 and 1 to q_1, the output function O maps q_5 to 1 and all states in $\{q_0, \ldots, q_4\}$ to 0, and the transition function $\delta(q_i, q_j)$ is defined as follows:

1. if $i + j < 5$, then the result is (q_{i+j}, q_0), and
2. if $i + j \geq 5$, then the result is (q_5, q_5). □

Exercise 1 Assume that all agents may err in two different ways. One possibility is that they do not apply the input function correctly and another is that they do not apply the transition function correctly. Fortunately, all are equipped with a special mechanism that automatically overwrites the faulty state with state $r \in \{r_1, r_2\}$, where r_1 and r_2 are the error reports/identifiers for the input function and the transition function, respectively. Try to adapt the "*flock of birds*" protocol to this scenario by keeping in mind that we require the protocol to give the correct output or report all the errors that have occurred.

Exercise 2 All birds in the flock are now additionally equipped with a sensor that determines their color, which is either black or white. Try to modify the "*flock of birds*" protocol in order to determine whether at least three black birds in the flock have elevated temperatures. Also exploit the white birds in order to (possibly[1]) improve the performance.
Hint: assume that the input symbols are of the form (i, j) where i corresponds to the temperature and j to the color.

5.2.2 *Stable Computation*

Assume a fair scheduler that keeps working forever and a protocol $\mathcal{A}$ that runs on a communication graph $G = (V, E)$. As already said, initially, each agent receives an input symbol from X. An *input assignment* $x : V \to X$ is a mapping specifying the input symbol of each agent in the population. Let $\mathcal{X} = X^V$ be the set of all possible input assignments, given the population V and the input alphabet X of $\mathcal{A}$. Population protocols, when controlled by infinitely working schedulers, do not halt. Instead of halting we require any computation of a protocol to *stabilize*. An output assignment $y : V \to Y$ is a mapping specifying the output symbol of each agent in the population. Any configuration $C \in \mathcal{C} = Q^V$ is associated with an output assignment $y_C = O \circ C$. A configuration C is said to be *output-stable* if for any configuration C' such that $C \overset{*}{\to} C'$ (any configuration reachable from C) $y_{C'} = y_C$. In words, a configuration C is output-stable if all agents maintain the output symbol that have under C in all subsequent steps, no matter how the scheduler proceeds thereafter. A computation $C_0, C_1, C_2, \ldots$ is *stable* if it contains an output-stable configuration C_i, where i is finite.

Definition 2 A population protocol $\mathcal{A}$ running on a communication graph $G = (V, E)$ *stably computes a predicate* $p : \mathcal{X} \to \{0, 1\}$, if, for any $x \in \mathcal{X}$, every computation of $\mathcal{A}$ on G beginning from $C_0 = I \circ x$ reaches in a finite number of steps

[1] We say *possibly*, because performance mainly depends on the scheduler. But if the scheduler is assumed to be probabilistic, then exploiting all agents should improve expected performance.

an output-stable configuration C_{stable} such that $y_{C_{\text{stable}}}(u) = p(x)$ for all $u \in V$. A predicate is *stably computable* if some population protocol stably computes it.

Assume, for example, that a computation of $\mathcal{A}$ on G begins from the initial configuration corresponding to an input assignment x. Assume, also, that $p(x) = 1$. If $\mathcal{A}$ stably computes p, then we know that after a finite number of steps (if, of course, the scheduler is fair) all agents will give 1 as output, and will continue doing so for ever. This means that if we wait for a sufficient, but finite, number of steps we can obtain the correct answer of p with input x by querying any agent in the population.

Definition 3 The *basic population protocol model* (or *standard*) assumes that the communication graph G is always directed and complete.

In the case of the basic model, a configuration simplifies to a vector of nonnegative integers that sum up to n indexed by states, and similarly for input assignments. Intuitively, we are allowed to do so because agents are anonymous and fairness guarantees that it does not matter in which agent each symbol or state lies. Moreover, here, stably computable predicates have to be *symmetric*. A predicate on input assignments p is called *symmetric* if for every $x = (\sigma_1, \sigma_2, \ldots, \sigma_n) \in \mathcal{X}$ and any x' which is a permutation of x's components, it holds that $p(x) = p(x')$ (in words, permuting the input symbols does not affect the predicate's outcome).

Thus, in the basic model, we can ignore the agents' underlying names to obtain a, seemingly, less descriptive, but sufficient for the basic model, definition of a configuration c as a $|Q|$-vector of nonnegative integers $(c_i)_{i=0,\ldots,|Q|-1}$, where $c_i = |c^{-1}(q_i)|$ and $|c^{-1}(q_i)|$ is equal to the number of agents to which state q_i is assigned by configuration c (the cardinality of the *preimage* of q_i), for all $i \in \{0, \ldots, |Q| - 1\}$. It is not hard to see that the above definition implies that $\sum_{i=0}^{|Q|-1} c_i = n$ for any configuration c.

Exercise 3 Do the same for the input assignments, that is, define formally their vector description.

Example 2 Now, that the most important notions have been defined, we are ready to prove that the "*flock of birds*" protocol stably computes the predicate

$$p(x) = \begin{cases} 1, \text{ if } x_1 >= 5 \\ 0, \text{ if } x_1 < 5 \end{cases}$$

where x_1 denotes the number of agents that get input symbol 1. Another way to write the predicate is $(x_1 \geq 5)$, which specifies that the value "true" is expected as output by all agents for every input assignment that provides at least five agents with the input symbol 1.

Proof There is no transition in δ that decreases the sum of the indices. In particular, if $i + j < 5$ then transitions are of the form $(q_i, q_j) \to (q_{i+j}, q_0)$ and leave the sum unaffected, while if $i + j \geq 5$ then transitions are of the form $(q_i, q_j) \to (q_5, q_5)$ and all strictly increase it except for $(q_5, q_5) \to (q_5, q_5)$ that leaves it unaffected. So the initial sum is always preserved except for the case where state q_5 appears. If

$x_1 < 5$ then it suffices to prove that state q_5 does not appear, because then all agents will forever remain in states $\{q_0, \ldots, q_4\}$ that give output 0. Assume that it appears. When this happened for the first time it was because the sum of the states of two interacting agents was at least 5. But this is a contradiction, because the initial sum should have been preserved until q_5 appeared. We now prove that if q_5 ever appears then all agents will eventually get it and remain to it forever. Obviously, if all get q_5 then they cannot escape from it, because no transition does this; thus, they forever remain to it. Now assume that q_5 has appeared in agent u and that agent $\upsilon \neq u$ never gets it. From the time that u got q_5 it could not change its state; thus any interaction of u and υ would make υ's state be q_5. This implies that u and υ did not interact for infinitely many steps, but this clearly violates the fairness condition (a configuration in which υ is in q_5 was always reachable in one step but was never reached). Now, if $x_1 \geq 5$ then it suffices to prove that q_5 appears. To see this, notice that all reachable configurations c for which $c_{q_5} = 0$ can reach in one step themselves and some configurations that preserve the sum but decrease the number of agents not in state q_0. Due to fairness, this will lead to a decrease by 1 in the number of non-q_0 agents in a finite number of steps, implying an increase in one agent's state index. This process ends either when all indices have been aggregated to one agent or when two agents, having a sum of indices at least 5, interact, and it must end, otherwise the number of q_0 agents would increase an unbounded number of times, being impossible for a fixed n. □

Note that such proofs are simplified a lot when we use arguments of the form "if q_5 appears then due to fairness all agents will eventually obtain it" and "due to fairness the sum will eventually be aggregated to one agent unless q_5 appears first" without getting into the details of the fairness assumption. Of course, we have to be very careful when using abstractions of this kind. □

Exercise 4 Consider the following protocol, known as "*parity protocol*":

> The input and output alphabets are $X = Y = \{0, 1\}$. The state of each agent consists of a data bit and a live bit. Initially, the data bit is equal to the input bit and the live bit is 1. For each state, the output bit is equal to the data bit. When two agents meet whose live bits are both 1, one sets its live bit to 0, and the other sets its data bit to the mod 2 sum of their data bits. When an agent with live bit 0 (a *sleeping* agent) meets an agent with live bit 1 (an *awake* agent), the former copies the data bit of the latter.

Prove that the "*parity protocol*" stably computes the predicate ($x_1 \bmod 2 = 1$), which is true iff there is an odd number of 1's in the input.

Exercise 5 Given a population protocol $\mathcal{A}$, if Q is the set of $\mathcal{A}$'s states and if $\mathcal{A}$ runs on the complete communication graph of n nodes (basic model), show that there are $\left(1 + \frac{n}{|Q|-1}\right)^{|Q|-1}$ different configurations.

Semilinear predicates are predicates whose support is a semilinear set. A *semilinear set* is the finite union of linear sets. A set of vectors in $\mathbb{N}^k$ is *linear* if it is of the form

$$\{\mathbf{b} + l_1\mathbf{a}_1 + l_2\mathbf{a}_2 + \cdots + l_m\mathbf{a}_m \mid l_i \in \mathbb{N}\},$$

where $\mathbf{b}$ is a *base* vector, $\mathbf{a}_i$ are *basis* vectors, and l_i are nonnegative integer coefficients. Moreover, semilinear predicates are precisely those predicates that can be defined by first-order logical formulas in *Presburger arithmetic*, as was proven by Ginsburg and Spanier [28].

Angluin *et al.* proved in [2] that any semilinear predicate is stably computable by the basic population protocol model and in [5] that any stably computable predicate, by the same model, is semilinear, thus together providing an exact characterization of the class of stably computable predicates:

Theorem 1 ([2, 5]) *A predicate is stably computable by the basic population protocol model iff it is semilinear.*

An immediate observation is that predicates like "the number of c's is the product of the number of a's and the number of b's (in the input assignment)" and "the number of 1's is a power of 2" are not stably computable by the basic model.

A *graph family*, or *graph universe*, is any set of communication graphs. Let $\mathcal{G}$ be a graph family. For any $G \in \mathcal{G}$, and given that X is the input alphabet of some protocol $\mathcal{A}$, there exists a set $\mathcal{X}_G$ of all input assignments *appropriate* for G, defined as $\mathcal{X}_G = X^{V(G)}$. Let now $\mathcal{X}_\mathcal{G} = \bigcup_{G\in\mathcal{G}}(\mathcal{X}_G \times \{G\})$ or, equivalently, $\mathcal{X}_\mathcal{G} = \{(x, G) \mid G \in \mathcal{G}$ and x is an input assignment appropriate for $G\}$. Then we have the following definition:

Definition 4 A population protocol $\mathcal{A}$ *stably computes a predicate* $p : \mathcal{X}_\mathcal{G} \to \{0, 1\}$ in a family of communication graphs $\mathcal{G}$, if, for any $G \in \mathcal{G}$ and any $x \in \mathcal{X}_G$, every computation of $\mathcal{A}$ on G beginning from $C_0 = I \circ x$ reaches in a finite number of steps an output-stable configuration C_{stable} such that $y_{C_{\text{stable}}}(u) = p(x, G)$ for all $u \in V(G)$.

Moreover, if p is a mapping from $\mathcal{G}$ to $\{0, 1\}$, that is, a *graph property* (obviously, independent of the input assignment), then we say that $\mathcal{A}$ stably computes property p.

Note that we can also consider undirected communication graphs. In the case of an undirected graph we only require that E is symmetric, but we keep the initiator–responder assumption. The latter is important to ensure deterministic transitions, since otherwise we would not be able to know which agent applies δ_1 and which δ_2.

5.3 Mediated Population Protocols

Consider now the following question: "Is there a way to extend the population protocol model and obtain a stronger model, without violating the uniformity and anonymity properties"? As we shall, in this section, see, the answer is in the affirmative. Although the idea is simple, it provides us with a model with significantly more computational power and extra capabilities in comparison to the population protocol model. The main modification is to allow the edges of the communication graph to store states from a finite set, whose cardinality is independent of the population size. Two interacting agents read the corresponding edge's state and update it, according to a global transition function, by also taking into account their own states.

5.3.1 Formal Definition

Definition 5 A *mediated population protocol* (MPP) is a 12-tuple $(X, Y, Q, I, O, S, \iota, \omega, r, K, c, \delta)$, where X, Y, Q, S and K are all finite sets and

1. X is the *input alphabet*,
2. Y is the *output alphabet*,
3. Q is the set of *agent states*,
4. $I : X \to Q$ is the *agent input function*,
5. $O : Q \to Y$ is the *agent output function*,
6. S is the set of *edge states*,
7. $\iota : X \to S$ is the *edge input function*,
8. $\omega : S \to Y$ is the *edge output function*,
9. r is the *output instruction* (informing the output-viewer how to interpret the output of the protocol),
10. K is the totally ordered *cost set*,
11. $c : E \to K$ is the *cost function*, and
12. $\delta : Q \times Q \times K \times S \to Q \times Q \times K \times S$ is the *transition function*.

We assume that the cost remains the same after applying δ and so we omit specifying an output cost. If $\delta(q_i, q_j, x, s) = \left(q_i', q_j', s'\right)$ (which, according to our assumption, is equivalent to $\delta(q_i, q_j, x, s) = \left(q_i', q_j', x, s'\right)$), we call $(q_i, q_j, x, s) \to \left(q_i', q_j', s'\right)$ a *transition*, and we define $\delta_1(q_i, q_j, x, s) = q_i'$, $\delta_2(q_i, q_j, x, s) = q_j'$ and $\delta_3(q_i, q_j, x, s) = s'$. Here, we, additionally, call δ_3 the *edge acquisition* (after the corresponding interaction).

In most cases we assume that $K \subset \mathbb{Z}^+$ and that $c_{\max} = \max_{w \in K} \{w\} = \mathcal{O}(1)$. Generally, if $c_{\max} = \max_{w \in K} \{|w|\} = \mathcal{O}(1)$ then any agent is capable of storing at most k cumulative costs (at most the value $kc_{\max}$), for some $k = \mathcal{O}(1)$, and we say that the cost function is *useful* (note that a cost range that depends on the population size could make the agents incapable for even a single cost storage and any kind of optimization would be impossible).

A *network configuration* is a mapping $C : V \cup E \to Q \cup S$ specifying the agent state of each agent in the population and the edge state of each edge in the communication graph. Let C and C' be network configurations, and let u, υ be distinct agents. We say that C goes to C' via encounter $e = (u, \upsilon)$, denoted $C \overset{e}{\to} C'$, if

$$\begin{aligned}
C'(u) &= \delta_1(C(u), C(\upsilon), x, C(e)) \\
C'(\upsilon) &= \delta_2(C(u), C(\upsilon), x, C(e)) \\
C'(e) &= \delta_3(C(u), C(\upsilon), x, C(e)) \\
C'(z) &= C(z), \text{ for all } z \in (V - \{u, \upsilon\}) \cup (E - e).
\end{aligned}$$

The definitions of *execution* and *computation* are the same as in the population protocol model but concern network configurations. Note that the mediated population protocol model *preserves both uniformity and anonymity* properties. As a result, any MPP's *code* is of *constant size* and, thus, can be stored in each agent (device) of the population.

A configuration C is called *r-stable* if one of the following conditions holds:

- If the problem concerns a subgraph to be found, then C should fix a subgraph that will not change in any C' reachable from C.
- If the problem concerns a function to be computed by the agents, then an r-stable configuration drops down to an output-stable configuration.

We say that a protocol $\mathcal{A}$ *stably solves* a problem Π, if for every instance I of Π and every computation of $\mathcal{A}$ on I, the network reaches an r-stable configuration C that gives the correct solution for I if interpreted according to the output instruction r. If instead of a problem Π we have a function f to be computed, we say that $\mathcal{A}$ *stably computes* f.

In the special case where Π is an optimization problem, a protocol that stably solves Π is called an *optimizing population protocol* for problem Π.

Example 3 We will present now a MPP with a leader that stably solves the following problem:

Problem 1 (Transitive Closure) We are given a complete directed communication graph $G = (V, E)$. Let E' be a subset of E. For all $e \in E'$ it holds that initially the state of e is 1. We are asked to find the transitive closure of G', that is, find a new edge set E^* that will contain a directed edge (u, υ) joining any nodes u, υ for which there is a non null path from u to υ in G' (note that always $E' \subseteq E^*$).

We assume a *controlled input assignment* $W : E \rightarrow X$ that allows us to give input 1 to any edge belonging to E' and input 0 to any other edge. Moreover, we assume that initially all agents are in state q_0, except for a unique leader that is in state l. □

The MPP $TranClos$ (Protocol 3) stably solves the transitive closure problem (Problem 1). You are asked to prove this in Exercise 6. Let us, first, explain one after the other the protocol's components before explaining its functionality (that is, what is done by the transition function δ). Both the input alphabet X and the output alphabet Y are binary, that is, both consist of the symbols 0 and 1. The set of agent states Q consists of the states l, q_0, q_1, q_1', q_2, q_2', and q_3, which are the states that

Protocol 3 *TranClos*

```
1: $X = Y = \{0, 1\}$
2: $Q = \{l, q_0, q_1, q_1', q_2, q_2', q_3\}$
3: $S = \{0, 1\}$
4: controlled input assignment: "$W(e') = 1$, for all $e' \in E'$, and $W(e) = 0$, for all $e \in E - E'$"
5: $\iota(x) = x$, for all $x \in X$
6: $\omega(s) = s$, for all $s \in S$
7: $r$: "Collect the subgraph induced by all $e \in E$ for which $\omega(s_e) = 1$ (where $s_e$ is the state of
   e)"
8: $\delta$:
```

$$
\begin{array}{ll}
(l, q_0, 0) \to (q_0, l, 0) & (q_2, q_0, 1) \to \left(q_2', q_3, 1\right) \\
(l, q_0, 1) \to (q_1, q_2, 1) & (q_1, q_3, x) \to \left(q_1', q_0, 1\right), \text{ for } x \in \{0, 1\} \\
(q_1, q_2, 1) \to (q_0, l, 1) & \left(q_1', q_2', 1\right) \to (q_0, l, 1)
\end{array}
$$

agents may obtain during the protocol's computations. The set of edge states S is binary, which means that the edges joining the agents will at any time be in one of the states 0 and 1. The controlled input assignment simply specifies that all edges belonging to E' are initially in state 1 (by taking into account that $\iota(x) = x$, for all $x \in X$) and all remaining edges of the communication graph are initially in state 0. This is done in order to help the protocol distinguish E'. $\omega(s) = s$, for all $s \in S$, simply says that the output of any edge is its current state, thus, either 0 or 1. Finally, the output instruction r informs the user that the protocol's output will consist of all edges that will eventually output the value 1. In this case, these edges will form the transitive closure of the communication graph G. We next discuss the protocol's functionality, which is described by the transition function δ.

The protocol *TranClos* (Protocol 3) repeats the following procedure. Initially, by assumption, there is a unique leader u in state l and all the other agents are in q_0. When the leader u interacts with an agent υ in q_0 through (u, υ) in state 0, the agents swap states, that is, now υ is the unique leader. If, instead, (u, υ) is in state 1, then the leader gets q_1 and υ gets q_2. After the latter has happened, all agents are in q_0 except for u and υ which are in q_1 and q_2, respectively, while (u, υ) is in state 1, and only the rules $(q_1, q_2, 1) \to (q_0, l, 1)$ and $(q_2, q_0, 1) \to \left(q_2', q_3, 1\right)$ can apply (all the other rules have no effect). If the former applies first, then the population goes to a configuration similar to the initial one, with a unique leader and all the other agents in q_0. This rule is important (although maybe not obvious why) since it guarantees that, if υ, which is in q_2, has no outgoing neighbor w, where $q_w = q_0$ and $s_{(\upsilon,w)} = 1$, then the protocol will not get stuck. If, instead, the latter applies first, then υ has interacted with an agent w in q_0 where (υ, w) is in state 1. Now υ gets q_2' and w gets q_3. After this step, the protocol has formed the directed path $u\upsilon w$, with agent states q_1, q_2', q_3, respectively, and (u, υ), (υ, w) (i.e., the edges of the path) in state 1. From now on, only $(q_1, q_3, x) \to \left(q_1', q_0, 1\right)$ can apply, which simply assigns state 1 to the edge (u, w). Finally, the population remains again with a unique leader, υ, and all the other agents in q_0, simply proceeding with the same general operation that we have just described.

Exercise 6 Give a formal proof that the MPP *TranClos* (Protocol 3) stably solves the transitive closure problem (Problem 1).

Exercise 7 Assume that the input symbols are only 0 and 1 and that the communication graph $G = (V, E)$ is any directed graph. Let G' be the subgraph of G induced by $V' = \{u \in V \mid u \text{ gets the input symbol } 1\}$. Devise a MPP that will construct a (not necessarily connected) subgraph $G'' = (V'', E'')$ of G', in which all nodes have in degree at most 1 and out degree at most 1.

5.3.2 Computational Power

The population protocol model is a special case of the mediated population protocol model (try to prove it). Moreover, as we shall see, there exists a MPP protocol that stably computes the non semilinear predicate $(N_c = N_a \cdot N_b)$. In words, it eventually decides whether the number of c's in the input assignment is equal to the product of the number of a's and the number of b's. The following definitions will prove useful for our purpose.

Definition 6 A MPP $\mathcal{A}$ has *stabilizing states* if in any computation of $\mathcal{A}$, after a finite number of interactions, the states of all agents stop changing.

Definition 7 We say that a predicate is *strongly stably computable* by the MPP model if it is stably computable with the predicate output convention, that is, all agents eventually agree on the correct output value.

Protocol 4 kkk*VarProduct*

```
1: $X = \{a, b, c, 0\}$
2: $Y = \{0, 1\}$
3: $Q = \{a, \dot{a}, b, c, \bar{c}, 0\}$
4: $S = \{0, 1\}$
5: $I(x) = x$, for all $x \in X$
6: $O(a) = O(b) = O(\bar{c}) = O(0) = 1$ and $O(c) = O(\dot{a}) = 0$
7: $\iota(x) = 0$, for all $x \in X$
8: $r$: "If there is at least one agent with output 0, reject; otherwise, accept."
9: $\delta$: $(a, b, 0) \to (\dot{a}, b, 1)$, $(c, \dot{a}, 0) \to (\bar{c}, a, 0)$, $(\dot{a}, c, 0) \to (a, \bar{c}, 0)$
```

Theorem 2 *The MPP* VarProduct *(Protocol 4) stably computes (according to the output instruction r) the predicate $(N_c = N_a \cdot N_b)$ in the family of complete directed communication graphs.*

Proof Notice that the number of links leading from agents in state a to agents in state b equals $N_a \cdot N_b$. For each a the protocol tries to erase b c's. Each a is able to remember the b's that it has already counted (for every such b a c has been erased) by marking the corresponding links. If the c's are less than the product then at least one $\dot{a}$ remains and if the c's are more at least one c remains. In both cases at least one agent that outputs 0 remains. If $N_c = N_a \cdot N_b$ then every agent eventually outputs 1. □

Theorem 3 *Let p be any predicate on input assignments. Let $\mathcal{A}$ be a MPP that stably computes p with stabilizing states in some family of directed and connected communication graphs $\mathcal{G}$ and also assume that $\mathcal{A}$ contains an instruction r that defines a semilinear predicate t on multisets of $\mathcal{A}$'s agent states. Since t is semilinear, it is stably computable with stabilizing inputs by the PP model [1, 5], and, thus, by the MPP model. Let $\mathcal{B}$ be a MPP that strongly stably computes t with stabilizing inputs in $\mathcal{G}$.*

If all the above hold then $\mathcal{A}$ can be composed with $\mathcal{B}$ to give a new MPP $\mathcal{C}$ satisfying the following properties:

- *$\mathcal{C}$ is formed by the composition of $\mathcal{A}$ and $\mathcal{B}$,*
- *its input is $\mathcal{A}$'s input,*
- *its output is $\mathcal{B}$'s output, and*
- *$\mathcal{C}$ strongly stably computes p (i.e., all agents agree on the correct output) in $\mathcal{G}$.*

Exercise 8 Prove Theorem 3.
Hint: $\mathcal{B}$ will make use of the stabilizing inputs idea from [1]; its inputs that eventually stabilize are $\mathcal{A}$'s states.

Theorems 2 and 3 together imply that the MPP model strongly stably computes *VarProduct* which is non semilinear. Since the MPP model strongly stably computes a non semilinear predicate and the PP model is a special case of MPP, it follows that the class of computable predicates by MPP is a proper superset of the class of computable predicates by PP. In other words, *the MPP model is computationally stronger than the PP model*.

In what concerns the class of stably computable predicates by MPP, recent (unpublished) research shows that it is a superset of *SSPACE(n)* (symmetric predicates in *LINSPACE*). We also know the following upper bound: "Any predicate that is stably computable by the MPP model in any family of communication graphs belongs to the space complexity class *NSPACE(m)*" (recall that $m = |E|$). The idea is simple: By using the MPP that stably computes the predicate we construct a nondeterministic Turing Machine that guesses in each step the next selection of the scheduler (thus the next configuration). The machine always replaces the current configuration with a new legal one, and, since any configuration can be represented explicitly with $\mathcal{O}(m)$ space, any branch uses $\mathcal{O}(m)$ space. The machine accepts if some branch reaches a configuration C that satisfies instruction r of the protocol, and if, moreover, no configuration reachable from C violates r (i.e., C must also be r-stable).

5.4 The GDM Model

Here we deal with MPP's ability to decide graph languages. To do so, we consider a special case of the mediated population protocol model, the *Graph Decision Mediated population protocol model*, or simply *GDM* model.

5.4.1 Formal Definition

Definition 8 A *GDM* protocol is an 8-tuple $(Y, Q, O, S, r, \delta, q_0, s_0)$, where Y, Q, and S are all finite sets and

1. $Y = \{0, 1\}$ is the *binary output alphabet*,
2. Q is the set of *agent states*,
3. $O : Q \to Y$ is the *agent output function*,
4. S is the set of *edge states*,
5. r is the *output instruction*,
6. $\delta : Q \times Q \times S \to Q \times Q \times S$ is the *transition function*,
7. $q_0 \in Q$ is the *initial agent state*, and
8. $s_0 \in S$ is the *initial edge state*.

If $\delta(a, b, s) = (a', b', s')$, we call $(a, b, s) \to (a', b', s')$ a *transition* and we define $\delta_1(a, b, s) = a'$, $\delta_2(a, b, s) = b'$, and $\delta_3(a, b, s) = s'$.

Let $\mathcal{U}$ be a graph universe. A *graph language* L is a subset of $\mathcal{U}$ containing communication graphs that possibly share some common property. For example, a common graph universe is the set of all possible directed and weakly connected communication graphs, denoted by $\mathcal{G}$, and $L = \{G \in \mathcal{G} \mid G$ has an even number of edges$\}$ is a possible graph language w.r.t. $\mathcal{G}$.

A GDM protocol may run on any graph from a specified graph universe. The graph on which the protocol runs is considered as the *input graph* of the protocol. Note that GDM protocols have no sensed input. Instead, we require each agent in the population to be initially in the initial agent state q_0 and each edge of the communication graph to be initially in the initial edge state s_0. In other words, the initial network configuration, C_0, of any GDM protocol is defined as $C_0(u) = q_0$, for all $u \in V$, and $C_0(e) = s_0$, for all $e \in E$, and any input graph $G = (V, E)$.

We say that a GDM protocol $\mathcal{A}$ *accepts* an input graph G if in any computation of $\mathcal{A}$ on G after finitely many interactions all agents output the value 1 and continue doing so in all subsequent (infinite) computational steps. By replacing 1 with 0 we get the definition of the *reject* case.

Definition 9 We say that a GDM protocol $\mathcal{A}$ *decides* a graph language $L \subseteq \mathcal{U}$ if it accepts any $G \in L$ and rejects any $G \notin L$.

Definition 10 A graph language is said to be *GDM-decidable*, or simply *decidable*, if some GDM protocol decides it.

5.4.2 Weakly Connected Graphs

5.4.2.1 Decidability

The most meaningful graph universe is $\mathcal{G}$ containing all possible directed and weakly connected communication graphs, without self-loops or multiple edges, of any finite number of nodes greater than or equal to 2 (we do not allow the empty graph, the graph with a unique node and we neither allow infinite graphs). Here the graph universe is $\mathcal{G}$ and, thus, a graph language can only be a subset of $\mathcal{G}$ (moreover, its elements must share some common property).

We begin with some easy to prove, but often useful, closure results.

Theorem 4 *The class of decidable graph languages is closed under complement, union, and intersection operations.*

Proof First we show that for any decidable graph language L its complement $\overline{L}$ is also decidable. From the definition of decidability there exists a GDM protocol $\mathcal{A}_L$ that decides L. Thus, for any $G \in \mathcal{G}$ and any computation of $\mathcal{A}_L$ on G all agents eventually output 1 if $G \in L$ and 0 otherwise. By complementing the output map $O_{\mathcal{A}}$ of $\mathcal{A}$ we obtain a new protocol $\overline{\mathcal{A}}$, with output map defined as $O_{\overline{\mathcal{A}}}(q) = 1$ iff $O_{\mathcal{A}}(q) = 0$, for all $q \in Q_{\mathcal{A}} = Q_{\overline{\mathcal{A}}}$, whose agents eventually output 1 if $G \notin L$ and 0 otherwise, thus deciding $\overline{L}$.

Now we show that for any decidable graph languages L_1 and L_2, $L_3 = L_1 \cup L_2$ is also decidable. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be GDM protocols that decide L_1 and L_2, respectively (we know their existence). We let the two protocols operate in parallel, i.e., we devise a new protocol $\mathcal{A}_3$ whose agent and edge states consist of two components, one for protocol $\mathcal{A}_1$ and one for $\mathcal{A}_2$. Let O_1 and O_2 be the output maps of the two protocols. We define the output map O_3 of $\mathcal{A}_3$ as $O_3(q, q') = 1$ iff at least one of $O_1(q)$ and $O_2(q')$ equals to 1, for all $q \in Q_{\mathcal{A}_1}$ and $q' \in Q_{\mathcal{A}_2}$. If $G \in L_3$ then at least one of the two protocols has eventually all its agent components giving output 1, thus $\mathcal{A}_3$ correctly answers "accept", while if $G \notin L_3$ then both protocols have eventually all their agent components giving output 0, thus $\mathcal{A}_3$ correctly answers "reject". We conclude that $\mathcal{A}_3$ decides L_3 which proves that L_3 is decidable.

By defining the output map O_3 of $\mathcal{A}_3$ as $O_3(q, q') = 1$ iff $O_1(q) = O_2(q') = 1$, for all $q \in Q_{\mathcal{A}_1}$ and $q' \in Q_{\mathcal{A}_2}$, and making the same composition as before, it is easy to see that in this case $\mathcal{A}_3$ decides the intersection of L_1 and L_2.

Note, however, that in each union and intersection operation the resulting protocol's size is the product of the sizes of the composed protocols. It follows that the closure under these two operations can only hold for a constant number of subsequent applications. □

Example 4 Let us now illustrate what we have seen so far by presenting a parametric GDM protocol that decides the graph language $N_k^{\text{out}} = \{G \in \mathcal{G} \mid G$ has some node with at least k outgoing neighbors$\}$ for any $k = \mathcal{O}(1)$.

We provide a high-level description of the protocol. Initially all agents are in q_0 and all edges in 0. The set of agent states is $Q = \{q_0, \dots, q_k\}$, the set of edge states is binary and the output function is defined as $O(q_k) = 1$ and $O(q_i) = 0$ for all $i \in \{0, \dots, k-1\}$. We now describe the transition function. In any interaction through an edge in state 0, the initiator visits an unvisited outgoing edge, so it marks it by updating the edge's state to 1 and increases its own state index by one, e.g., initially $(q_0, q_0, 0)$ yields $(q_1, q_0, 1)$, and, generally $(q_i, q_j, 0) \to (q_{i+1}, q_j, 1)$, if $i+1 < k$ and $j < k$, and $(q_i, q_j, 0) \to (q_k, q_k, 1)$, otherwise. Whenever two agents meet through a marked edge they do nothing, except for the case where only one of them is in the special alert state q_k. If the latter holds, then both go to the alert state, since in this case the protocol has located an agent with at least k outgoing neighbors. To conclude, all agents count their outgoing edges and initially output 0. Iff one of them marks its kth outgoing edge, both end points of that edge go to an alert state q_k that is eventually propagated to the whole population and whose output is 1, indicating that G belongs to N_k^{out}. Clearly, the described protocol decides N_k^{out}, which means that N_k^{out} is a decidable graph language. Moreover, the same must hold for $\overline{N}_k^{\text{out}}$ because, according to Theorem 4, the class of decidable graph languages is closed under complement. Note that $\overline{N}_k^{\text{out}}$ contains all graphs that have no node with at least $k = \mathcal{O}(1)$ outgoing neighbors. In other words, the GDM model can decide if all nodes have less than k outgoing edges, which is simply the well-known bounded by k out degree predicate. □

Example 5 We show now that the graph language $P_k = \{G \in \mathcal{G} \mid G$ has at least one directed path of at least k edges$\}$ is decidable for any $k = \mathcal{O}(1)$ (the same holds for $\overline{P}_k$).

If $k = 1$ the protocol that decides P_1 is trivial, since it accepts iff at least one interaction happens (in fact it can always accept since all graphs have at least two nodes and they are weakly connected, and thus $P_1 = \mathcal{G}$). The protocol *DirPath* (Protocol 5) that we have constructed decides P_k for any constant $k > 1$.

Protocol 5 *DirPath*

```
1: $Q = \{q_0, q_1, 1, \dots, k\}$
2: $S = \{0, 1\}$
3: $O(k) = 1$, $O(q) = 0$, for all $q \in Q - \{k\}$
4: $r$: "Get any $u \in V$ and read its output"
5: $\delta$:
```

$$\begin{aligned}
(q_0, q_0, 0) &\to (q_1, 1, 1)\\
(q_1, x, 1) &\to (x-1, q_0, 0),\ \text{if } x \geq 2\\
&\to (q_0, q_0, 0),\ \text{if } x = 1\\
(x, q_0, 0) &\to (q_1, x+1, 1),\ \text{if } x+1 < k\\
&\to (k, k, 0),\ \text{if } x+1 = k\\
(k, \cdot, \cdot) &\to (k, k, \cdot)\\
(\cdot, k, \cdot) &\to (k, k, \cdot)
\end{aligned}$$

Initially all nodes are in q_0 and all edges in 0. The protocol tries to expand disjoint paths. When rule $(q_0, q_0, 0) \to (q_1, 1, 1)$ applies, the initiator goes to q_1 indicating that it is a node of an active path, the responder goes to 1 indicating that it is the head of an active path of length 1, and the edge goes to 1 indicating that it is part of an active path. By inspecting the transition function it is easy to see that the nodes of two disjoint active paths have no way of interacting with each other (in fact, the interactions happening between them leave their interacting components unaffected). This holds because all nodes in q_1 do nothing when communicating through an edge in state 0 and disjoint active paths can only communicate through edges in state 0. Moreover, the heads of the paths only expand by communicating with nodes in q_0 which, of course, cannot be nodes of active paths (all nodes of active paths are in q_1 except for the heads which are in states from $\{1, \dots, k-1\}$). There are two main possibilities for an active path: either the protocol expands it, thus obtaining a node and an edge and increasing the head counter by one, or shrinks it, thus releasing a node and an edge and decreasing the head counter by one. Eventually, a path will either be totally released (all its nodes and edges will return to the initial states) or it will become of length k. In the first case the protocol simply keeps working but in the second, a path of length at least k was found and state k that outputs 1 is correctly propagated. The crucial point is that state k is the only state that outputs 1 and can only be reached and propagated by the agents iff there exists some path of length at least k. Moreover, if such a path exists, due to fairness assumption, the protocol will eventually manage to find it. □

The following graph languages are also decidable by the GDM model:

1. $N_{\text{even}} = \{G \in \mathcal{G} \mid |V(G)| \text{ is even}\}$.
2. $E_{\text{even}} = \{G \in \mathcal{G} \mid |E(G)| \text{ is even}\}$.
3. $K_k^{\text{out}} = \{G \in \mathcal{G} \mid$ Any node in G has at least k outgoing neighbors$\}$ for any $k = \mathcal{O}(1)$.
4. $M_{\text{out}} = \{G \in \mathcal{G} \mid G$ has some node with more outgoing than incoming neighbors$\}$.

Of course, by closure under complement, the same holds for their complements.

Exercise 9 Do you think it is possible to construct a GDM protocol that decides the graph language consisting of all directed and weakly connected communication graphs in which all nodes have at most $k = \mathcal{O}(1)$ incoming edges and in which the number of nodes is at least 5% of the number of edges? If yes, construct the protocol and prove its correctness; if no, explain why.

5.4.2.2 Undecidability

If we allow only GDM protocols with stabilizing states, i.e., GDM protocols that in any computation after finitely many interactions stop changing their states, then we can prove that a specific graph language w.r.t. $\mathcal{G}$ is *undecidable*. In particular, we can prove that there exists no GDM protocol with stabilizing states to decide the graph language

$$2C = \{G \in \mathcal{G} \mid G \text{ has at least two nodes } u, \upsilon \text{ s.t. both } (u, \upsilon), (\upsilon, u) \in E(G) \text{ (in other words, } G \text{ has at least one 2-cycle)}\}.$$

The proof is based on the following lemma.

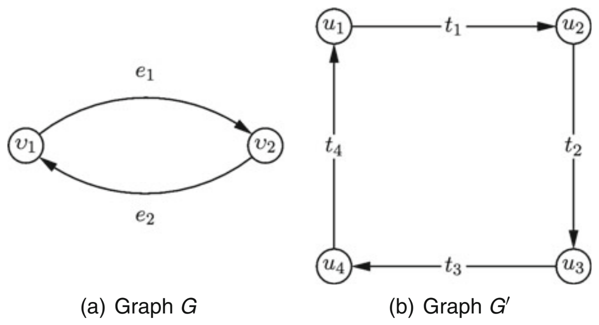


Fig. 5.1 $G \in 2C$ and $G' \notin 2C$

Lemma 1 *For any GDM protocol $\mathcal{A}$ and any computation $C_0, C_1, C_2, \ldots$ of $\mathcal{A}$ on G (Fig. 5.1a) there exists a computation $C'_0, C'_1, C'_2, \ldots, C'_i, \ldots$ of $\mathcal{A}$ on G' (Fig. 5.1b) s.t.*

$$\begin{aligned} C_i(\upsilon_1) &= C'_{2i}(u_1) = C'_{2i}(u_3) \\ C_i(\upsilon_2) &= C'_{2i}(u_2) = C'_{2i}(u_4) \\ C_i(e_1) &= C'_{2i}(t_1) = C'_{2i}(t_3) \\ C_i(e_2) &= C'_{2i}(t_2) = C'_{2i}(t_4) \end{aligned}$$

for any finite $i \geq 0$.

Exercise 10 Prove Lemma 1 by using induction on i.

Lemma 1 shows that if a GDM protocol $\mathcal{A}$ with stabilizing states could decide $2C$ then there would exist a computation of $\mathcal{A}$ on G' forcing all agents to output incorrectly the value 1 in finitely many steps. But G' does not belong to $2C$, and, since $\mathcal{A}$ decides $2C$, all agents must correct their states to eventually output 0. By taking into account the fact that $\mathcal{A}$ has stabilizing states it is easy to reach a contradiction and prove that no GDM protocol with stabilizing states can decide $2C$. Whether the graph language $2C$ is undecidable by the GDM model in the general case (not only by GDM protocols with stabilizing states) remains an interesting open problem.

5.4.3 All Possible Directed Graphs

It is not hard to show that if the graph universe, $\mathcal{H}$, is allowed to contain also disconnected communication graphs, then in this case the GDM model is incapable of deciding even a single nontrivial graph language (we call a graph language L *nontrivial* if $L \neq \emptyset$ and $L \neq \mathcal{H}$). Here we assume the graph universe $\mathcal{H}$ consisting of all possible directed communication graphs, without self-loops or multiple edges of any finite number of nodes greater or equal to 2 (we now also allow graphs that are not even weakly connected). So, now, a graph language can only be a subset of $\mathcal{H}$.

The crucial part is to show that for any nontrivial graph language L, there exists some disconnected graph G in L where at least one component of G does not belong to L or there exists some disconnected graph G' in $\overline{L}$ where at least one component of G' does not belong to $\overline{L}$ (or both). If the statement does not hold then any disconnected graph in L has all its components in L and any disconnected graph in $\overline{L}$ has all its components in $\overline{L}$.

1. All connected graphs belong to L. Then $\overline{L}$ contains at least one disconnected graph (since it is nontrivial) that has all its components in L, which contradicts the fact that the components of any disconnected graph in $\overline{L}$ also belong to $\overline{L}$.
2. All connected graphs belong to $\overline{L}$. The contradiction is symmetric to the previous case.
3. L and $\overline{L}$ contain connected graphs G and G', respectively. Their disjoint union $U = (V \cup V', E \cup E')$ is disconnected, belongs to L or $\overline{L}$ but one of its components belongs to L and the other to $\overline{L}$. The latter contradicts the fact that both components should belong to the same language.

Now it will not be hard to prove the impossibility result as an exercise.

Exercise 11 Prove that any nontrivial graph language $L \subset \mathcal{H}$ is undecidable by the GDM model.
Hint: notice that agents of different components cannot communicate with each other.

Exercise 12 Do you think that Connectivity property is GDM-decidable?

5.5 Community Protocols

In this section, we present the *Community Protocol* model, which was proposed by Guerraoui and Ruppert [29] and is another extension of the population protocol model. In fact, this, recently proposed, model makes the assumption that the agents are equipped with unique ids and are also allowed to store a fixed number of other agents' ids. The term "community" in the model's name is used to emphasize the fact that the agents here form a collection of unique individuals similar to the notion of a human community, in contrast to a population which is merely an agglomeration of nameless multitude.

5.5.1 The Model

As usual, we start with a formal definition of the model, and then a somewhat informal description of its functionality follows.

Definition 11 Let U be an infinite ordered set containing all possible ids. A *Community Protocol Algorithm* is an 8-tuple $(X, Y, B, d, I, O, Q, \delta)$, where X, Y, and B are all finite sets and

1. X is the *input alphabet*,
2. Y is the *output alphabet*,
3. B is the set of *basic states*,
4. d is a nonnegative integer representing the number of ids that can be remembered by an agent,
5. $I : X \to B$ is the *input function* mapping input symbols to basic states,
6. $O : B \to Y$ is the *output function* mapping basic states to outputs,
7. $Q = B \times (U \cup \{\perp\})^d$ is the set of *agent states*, and
8. $\delta : Q \times Q \to Q \times Q$ is the *transition function*.

If $\delta(a, b) = (a', b')$, we call $(a, b) \to (a', b')$ a *transition* and we define $\delta_1(a, b) = a'$ and $\delta_2(a, b) = b'$.

The first obvious difference between this and the population protocol model is that the agent states are allowed to contain up to d ids. Additionally, each agent is assumed to have its own unique id from the industry (which is an element of U). As in the population protocol model, initially each agent $i \in \{1, \dots, n\}$ receives an input symbol from X. Note that the ith agent is the agent whose id is in position i in the ascending ordering of agent ids. An input assignment $x \in \mathcal{X} = X^V$ is again any n-vector of input symbols, where x_i is the input to agent i. Moreover, let id_i denote the actual id of agent i and $b_i = I(x_i)$ (that is, the initial basic state of agent i). Then the *initial state* of each agent i is of the form $(b_i, id_i, \perp, \perp, \dots, \perp)$. Thus, initially, each agent i is in basic state b_i, contains its own unique id id_i in the first position of its list of ids, and the remaining list is filled with $d - 1$ repetitions of the symbol $\perp$.

A *configuration C* is a vector in Q^n of the form $C = (q_1, q_2, \dots, q_n)$, where q_i is simply the state of agent i for all $i \in \{1, \dots, n\}$. Thus, the *initial configuration* corresponding to input assignment x is $((b_i, id_i, \perp, \perp, \dots, \perp))_{i=1}^{|x|}$, where again $b_i = I(x_i)$ and id_i is the actual id of agent i. The notions of *execution*, *computation*, and *fairness* are defined in the same way as in the population protocol model. Moreover, we will call the community protocol model, in which the communication graph is directed and complete, *basic community protocol* model (like we did with the population protocol model). The scheduler choosing the interactions is again assumed to be *fair*.

The output of an agent at any step of the computation is the output of its basic state. For example, the output of an agent in state $(b_i, id_i, 1, 5, \dots, \perp)$ is $O(b_i) \in Y$. A community protocol algorithm for the basic model *stably computes* a function $f : X^{\geq 2} \to Y$, where $X^{\geq 2}$ denotes the set of all finite strings over X of length at least 2, if for any $x \in X^{\geq 2}$ and any assignment of the symbols in x to the nodes of the complete communication graph of $|x|$ nodes, all agents, independently of the fair computation followed, eventually stabilize to the output $f(x)$, that is, a configuration is reached under which all agents output $f(x)$ and continue doing so forever, no matter how the computation proceeds thereafter (such a configuration is, as usual, called an *output-stable* configuration).

As in population protocols, algorithms are *uniform* (but, clearly, not *anonymous*). The reason is that their description makes no assumption of the *community size* n; thus their functionality remains identical for all complete communication graphs. That is why the set of ids U is infinite. The suspicious reader would notice that if we do not impose further restrictions on the model then the agents can use their d slots to store arbitrary amounts of information (by exploiting the fact that U is defined to be infinite), which is artificial. To avoid this, we impose a *local knowledge* constraint, according to which agents can only store ids that they have learned from other agents via interactions with them. To formalize this, let $l(q)$ denote the set of different ids appearing in the list of ids of state q. If $\delta(q_1, q_2) = (q_1', q_2')$ and $id \in l(q_1') \cup l(q_2')$ then $id \in l(q_1) \cup l(q_2)$ (in words, no new ids appear in the outcome of an interaction).

Additionally, an *operational* constraint is imposed that allows no other operations except for comparisons to be performed on ids by the agents. This constraint is only imposed to keep the model minimal, because it turns out that, even in the presence of this constraint, the model is surprisingly strong (computationally). Intuitively, if $((b_1, \ldots), (b_2, \ldots)) \to ((b_1', \ldots), (b_2', \ldots))$ is a transition in δ, then any transition with precisely the same basic states in which the ids of the lhs are replaced by ids that preserve the order (which, according to the local knowledge constraint, implies that also the ids in the rhs will preserve the order) also belongs to δ. Since this may be a little subtle, another way to think of it is the following. All interactions that do not differ w.r.t. the basic states of the agents and whose lists of ids contain ids that preserve the order, provide the agents with the same new pair of basic states and with new lists of ids that do not different w.r.t. the order of ids.

To make this precise, let $\delta(q_1, q_2) = (q_1', q_2')$. Moreover, let $id_1 < id_2 < \cdots < id_k$ be all ids in $l(q_1) \cup l(q_2) \cup l(q_1') \cup l(q_2')$ and let $id_1' < id_2' < \cdots < id_k'$ be ids. If $\rho(q)$ is the state obtained from q by replacing all occurrences of each id id_i by id_i', then we require that $\delta(\rho(q_1), \rho(q_2)) = (\rho(q_1'), \rho(q_2'))$ also holds.

Example 6 Assume that $\delta((b_1, 1, 2, \bot), (b_2, 7, \bot, \bot)) = ((b_1', 1, 7, \bot), (b_2', 2, 2, 1))$. Then it holds that $\delta((b_1, 2, 5, \bot), (b_2, 8, \bot, \bot)) = ((b_1', 2, 8, \bot), (b_2', 5, 5, 2))$. The reason is that $1 < 2 < 7$ and $2 < 5 < 8$ and we have replaced 1 by 2, 2 by 5, and 7 by 8, thus preserving the order of ids. Generally, $\delta((b_1, id_1, id_2, \bot), (b_2, id_3, \bot, \bot)) = ((b_1', id_1, id_3, \bot), (b_2', id_2, id_2, id_1))$ must hold for all $id_1, id_2, id_3 \in U$, where $id_1 < id_2 < id_3$, for the same reason. □

Exercise 13 Consider a transition function δ and let $\delta(q_1, q_2) = (q_1', q_2')$ be any transition. Let b_q denote the basic state of state q, and $id_{q,j}$ the jth id in the id list of q. δ is defined as follows. If $b_{q_1} = b_{q_2}$ then nothing happens. If $b_{q_1} \neq b_{q_2}$ then

- If $id_{q_1,j} > id_{q_2,j}$ and $id_{q_1,j}, id_{q_2,j} \neq \bot$ for some $j \in \{2, \ldots, d\}$, then $id_{q_2',i} = \bot$ and $id_{q_1',i} = id_{q_2,i}$ for all $i \in \{2, \ldots, d\}$, and $b_{q_1'} = b_{q_2'} = b_{q_1}$.
- Else $id_{q_1',i} = \bot$ and $id_{q_2',i} = id_{q_1,i}$ for all $i \in \{2, \ldots, d\}$, and $b_{q_1'} = b_{q_2'} = b_{q_2}$.

Does δ satisfy the local knowledge and operational constraints? Support your answer with a formal proof.

5.5.2 Computational Power

The community protocol model turns out to be extremely strong in terms of its computational power. In fact it turns out that any symmetric predicate $p : X^{\geq 2} \to \{0, 1\}$ is (stably) computable by the basic community protocol model if and only if it belongs to *NSPACE*$(n \log n)$, where, as usual, n denotes the community size. The reason that we consider symmetric predicates is that the identifiers of the model cannot be used to order the input symbols; thus an algorithm's functionality in the basic model has to be identical for any permutation of the inputs w.r.t. to the agents' ordering.

Definition 12 Let *CP* denote the class of all symmetric predicates p that are stably computable by the basic community protocol model.

First of all, we prove that any stably computable symmetric predicate p is in *NSPACE* $(n \log n)$.

Theorem 5 ([29]) *CP is a subset of NSPACE*$(n \log n)$.

Proof We will construct a nondeterministic TM N that decides the language $L_p = \{x \in X^{\geq 2} \mid p(x) = 1\}$ (the *support* of p) using at most *NPSACE*$(n \log n)$ = *NPSACE*$(| < x > | \log | < x > |)$ cells on any branch of its computation. The reason that the latter equality holds is that the input of p consists of n input symbols, picked from the set X whose cardinality is independent of n. This means that for any input x to the machine N (any element of L_p) it holds that $| < x > | = \mathcal{O}(n)$, where n is the community size.

First of all, we make the following natural assumption: n agents have w.l.o.g. the unique ids $1, 2, \ldots, n$. This implies that each id occupies $\mathcal{O}(\log n)$ cells in a TM. Moreover, there are d id slots in an agent's state, and since d is independent of n again $\mathcal{O}(\log n)$ cells suffice to store the list of ids of any state. The cardinality of B is also independent of n; thus we conclude that $\mathcal{O}(\log n)$ cells suffice to store any state of Q. A configuration is simply a vector consisting of n states; thus a configuration will occupy $\mathcal{O}(n \log n)$ cells of memory storage.

To accept input x, N must verify two conditions: That there exists a configuration C reachable from $I(x)$ (that here denotes the initial configuration corresponding to x), in which all basic states output $p(x)$, and that there is no configuration C' reachable from C, in which some basic state does not output $p(x)$.

The first condition is verified by guessing and checking a sequence of configurations, starting from $I(x)$ and reaching such a C. N guesses a C_{i+1} each time, verifies that $C_i \to C_{i+1}$ (begins from $C_0 = I(x)$, i.e., $i = 0$) and, if so, replaces C_i by C_{i+1}, otherwise drops this C_{i+1}. The second condition is the complement of a similar reachability problem. But *NSPACE* is closed under complement for all space functions $\geq \log n$ (see Immerman–Szelepcsényi theorem [35]). Thus, by taking into account that only one configuration is kept at any step of any branch and that the size of any configuration is $\mathcal{O}(n \log n)$, we conclude that N decides L_p in $\mathcal{O}(n \log n)$ space. □

A Schönhage's *Storage Modification Machine* (SMM) is a kind of pointer machine (not a distributed system). Its memory stores a finite directed graph of constant out degree with a distinguished node called the *center*. The edges of the graph are called *pointers*. The edges out of each node are labeled by distinct *directions* drawn from a finite set Δ. For example, a reasonable implementation of Δ could use all nonnegative integers up to the maximum out degree in the graph minus one. Any string $x \in \Delta^*$ can be used as a reference to the node that is reached if we begin from the center and follow the pointers whose labels are indicated by the sequence of symbols in x. We denote the node indicated by $x \in \Delta^*$ by $p^*(x)$. The basic operations of an SMM allow the machine to create nodes, modify pointers, and follow paths of pointers. We now formalize the above description.

Definition 13 A *Nondeterministic Storage Modification Machine* (NSMM) is a 3-tuple (Σ, Δ, P), where Σ and Δ are both finite sets and

1. Σ is the *input alphabet*,
2. Δ is the set of *distinct directions*, and
3. P is the *program*, which is a finite list of instructions.

Inputs to the SMM are finite strings from Σ^*. Programs may use instructions of the following types:

- *new*: creates a node, makes it the center, and sets all its outgoing pointers to the old center.
- *recenter* x, where $x \in \Delta^+$: changes the center of the graph to $p^*(x)$.
- *set* $x\delta$ *to* y, where $x, y \in \Delta^*$ and $\delta \in \Delta$: changes the pointer of node $p^*(x)$ that is labeled by δ to point to node $p^*(y)$.
- *if* $x = y$ *then goto* l, where $x, y \in \Delta^*$: jumps to (program) line l if $p^*(x) = p^*(y)$.
- *input* $l_1, \ldots, l_r$, where $l_1, \ldots, l_r$ are (program) line numbers: consumes the next input symbol (if there is one) and jumps to line l_i if that symbol is σ_i.
- *output* o, where $o \in \{0, 1\}$: causes the machine to halt and output o.
- *choose* l_0, l_1, where l_0 and l_1 are line numbers: causes the machine to transfer control either to line l_0 or to line l_1 nondeterministically.

When a node becomes unreachable from the center, it can be dropped from the graph, since it plays no further role in the computation. Space complexity is measured by the maximum number of (reachable) nodes present at any step of any branch of the machine's nondeterministic computation.

It can be proved that any language decided by a nondeterministic Turing Machine using $\mathcal{O}(S \log S)$ space can be decided by an NSMM using S nodes. Thus, to prove that all symmetric predicates in *NSPACE*$(n \log n)$ also belong to *CP* it suffices to show that there exists a community protocol that simulates an NSMM that uses $\mathcal{O}(n)$ nodes. The latter can be shown but, unfortunately, the construction is quite involved, so we skip it. Now by taking into account Theorem 5 we get the following exact characterization.

Theorem 6 ([29]) *CP is equal to the class of all symmetric predicates in NSPACE*$(n \log n)$.

5.6 Logarithmic Space Machines

In this section, we study another recently proposed model, called the *PALOMA* model [14]. In fact, it is a model of PAssively mobile MAchines (that we keep calling agents) equipped with two-way communication and each having a memory whose size is LOgarithmic in the population size n.

The reason for studying such an extension is that having *logarithmic communicating machines* seems to be more natural than communicating automata of constant memory. First of all, the *communicating machines* assumption is perfectly consistent with current technology (cellphones, iPods, PDAs, and so on). Moreover, *logarithmic* is, in fact, *extremely small*. For a convincing example, it suffices to mention that for a population consisting of 2^{266} agents, which is a number greater than the number of atoms in the observable universe, we only require each agent to have 266 cells of memory (while small-sized flash memory cards nowadays exceed 16GB of storage capacity)! Interestingly, it turns out that the agents, by assigning unique ids to themselves, are able to get organized into a distributed nondeterministic TM that makes full use of the agents' memories! The TM draws its nondeterminism by the nondeterminism inherent in the interaction pattern. It is here like the nameless multitude can turn itself into a well-organized community.

Definition 14 A *PALOMA* protocol $\mathcal{A}$ is a 7-tuple $(\Sigma, X, \Gamma, Q, \delta, \gamma, q_0)$, where Σ, X, Γ, and Q are all finite sets and

1. Σ is the *input alphabet*, where $\#, \sqcup \notin \Sigma$,
2. $X \subseteq \Sigma^*$ is the *set of input strings*,
3. Γ is the *tape alphabet*, where $\#, \sqcup \in \Gamma$ and $\Sigma \subset \Gamma$,
4. Q is the set of *states*,
5. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\} \times \{0, 1\}$ is the *internal transition function*,
6. $\gamma : Q \times Q \to Q \times Q$ is the *external transition function* (or *interaction transition function*), and
7. $q_0 \in Q$ is the *initial state*.

Each agent is equipped with the following:

- A *sensor* in order to sense its environment and receive a piece of the input (which is an input string from X).
- A *tape* (memory) consisting of $\mathcal{O}(\log n)$ cells. The tape is partitioned into three parts each consisting of $\mathcal{O}(\log n)$ cells: the leftmost part is the *working tape*, the middle part is the *output tape*, and the rightmost part is the *message tape* (we call the parts "tapes" because such a partition is equivalent to a 3-tape machine).

The last cell of each part contains permanently the symbol # (we assume that the machine never alters it); it is the symbol used to separate the three tapes and to mark the end of the overall tape.

- A *control unit* that contains the state of the agent and applies the transition functions.
- A *head* that reads from and writes to the cells and can move one step at a time, either to the left or to the right.
- A binary *working flag* either set to 1 meaning that the agent is *working* internally or to 0 meaning that the agent is *ready* for interaction.

Initially, all agents are in state q_0 and all their cells contain the *blank symbol* $\sqcup$ except for the last cell of the working, output, and message tapes that contain the *separator* #. We assume that all agents concurrently receive their sensed input (different agents may sense different data) as a response to a global start signal. The input is a string from X and after reception (or, alternatively, during reception, in an online fashion) it is written symbol by symbol on their working tape beginning from the leftmost cell. During this process the working flag is set to 1 and remains to 1 when this process ends (the agent may set it to 0 in future steps).

When its working flag is set to 1 we can think of the agent working as a usual Turing Machine (but it additionally writes the working flag). In particular, whenever the working flag is set to 1 the internal transition function δ is applied, the control unit reads the symbol under the head and its own state and updates its state and the symbol under the head, moves the head one step to the left or to the right and sets the working flag to 0 or 1, according to the internal transition function.

We assume that the set of states Q and the tape alphabet Γ, are both sets whose size is fixed and independent of the population size (i.e., $|Q| = |\Gamma| = \mathcal{O}(1)$); thus, there is, clearly, enough room in the memory of an agent to store both the internal and the external transition functions.

Again here, a *fair adversary scheduler* selects ordered pairs of agents to interact. Assume now that two agents u and υ are about to interact with u being the *initiator* of the interaction and υ being the *responder*. Let $f : V \rightarrow \{0, 1\}$ be a function returning the current value of each agent's working flag. If at least one of $f(u)$ and $f(\upsilon)$ is equal to 1, then nothing happens, because at least one agent is still working internally. Otherwise ($f(u) = f(\upsilon) = 0$), both agents are ready and an *interaction* is established. In the latter case, the external transition function γ is applied, the states of the agents are updated accordingly, the message of the initiator is copied to the message tape of the responder (replacing its contents) and vice versa (the real mechanism would require that each one receives the other's message and then copies it to its memory, because instant replacement would make them lose their own message, but this can be easily implemented with $\mathcal{O}(\log n)$ extra cells of memory, so it is not an issue), and finally the working flags of both agents are again set to 1.

Since each agent is a TM (of logarithmic memory), we use the notion of a configuration to capture its "state". An *agent configuration* is a quadruple (q, l, r, f), where $q \in Q$, $l, r \in \Gamma^{\mathcal{O}(\log n)} = \{s \in \Gamma^* \mid |s| = \mathcal{O}(\log n)\}$, and $f \in \{0, 1\}$.

q is the state of the control unit, l is the string to the left of the head (including the symbol scanned), r is the string to the right of the head, and f is the working flag indicating whether the agent is ready to interact ($f = 0$) or carrying out some internal computation ($f = 1$). Let $\mathcal{B}$ be the set of all agent configurations. Given two agent configurations $A, A' \in \mathcal{B}$, we say that A *yields* A' if A' follows A by a single application of δ.

A *population configuration* is a mapping $C : V \to \mathcal{B}$, specifying the agent configuration of each agent in the population. Let C, C' be population configurations and let $u \in V$. We say that C *yields* C' via *agent transition* u, denoted $C \overset{u}{\to} C'$, if $C(u)$ yields $C'(u)$ and $C'(w) = C(w), \forall w \in V - \{u\}$.

Let $q(A)$ denote the state of an agent configuration A, $l(A)$ its string to the left of the head including the symbol under the head, $r(A)$ its string to the right of the head, and $f(A)$ its working flag. Given two population configurations C and C', we say that C *yields* C' via *encounter* $e = (u, \upsilon) \in E$, denoted $C \overset{e}{\to} C'$, if one of the following two cases holds:

Case 1:

- $f(C(u)) = f(C(\upsilon)) = 0$ which guarantees that both agents u and υ are ready for interaction under the population configuration C.
- $r(C(u))$ and $r(C(\upsilon))$ are precisely the message strings of u and υ, respectively (this is a simplifying assumption stating that when an agent is ready to interact its head is over the last # symbol, just before the message tape),
- $C'(u) = (\gamma_1(q(C(u)), q(C(\upsilon))), l(C(u)), r(C(\upsilon)), 1)$,
- $C'(\upsilon) = (\gamma_2(q(C(u)), q(C(\upsilon))), l(C(\upsilon)), r(C(u)), 1)$, and
- $C'(w) = C(w), \forall w \in V - \{u, \upsilon\}$.

Case 2:

- $f(C(u)) = 1$ or $f(C(\upsilon)) = 1$, which means that at least one agent between u and υ is working internally under the population configuration C, and
- $C'(w) = C(w), \forall w \in V$. In this case no effective interaction takes place, thus the population configuration remains the same.

Generally, we say that C *yields* (or *can go in one step to*) C', and write $C \to C'$, if $C \overset{e}{\to} C'$ for some $e \in E$ (via encounter) or $C \overset{u}{\to} C'$ for some $u \in V$ (via agent transition), or both. We say that C' is *reachable* from C, and write $C \overset{*}{\to} C'$ if there is a sequence of population configurations $C = C_0, C_1, \ldots, C_t = C'$ such that $C_i \to C_{i+1}$ holds for all $i \in \{0, 1, \ldots, t-1\}$. An *execution* is a finite or infinite sequence of population configurations $C_0, C_1 \ldots$, so that $C_i \to C_{i+1}$. An infinite execution is *fair* if for all population configurations C, C' such that $C \to C'$, if C appears infinitely often then so does C'. A *computation* is an infinite fair execution.

Note that the PALOMA model *preserves uniformity*, because X, Γ, and Q are all finite sets whose cardinality is independent of the population size. Thus, protocol descriptions have also no dependence on the population size. Moreover, PALOMA protocols are *anonymous*, since initially all agents are identical and have no unique identifiers.

Example 7 We present now a PALOMA protocol that stably computes the predicate $(N_c = N_a \cdot N_b)$ (on the complete communication graph of n nodes) that is, all agents eventually decide whether the number of cs in the input assignment is the product of the number of as and the number of bs. We give a high-level description of the protocol.

Initially, all agents have one of a, b, and c written on the first cell of their working memory (according to their sensed value). That is, the set of input strings is $X = \Sigma = \{a, b, c\}$. Each agent that receives input a goes to state a and becomes ready for interaction (sets its working flag to 0). Agents both in state a and b do nothing when interacting with agents in state a and agents in state b. An agent in c initially creates in its working memory three binary counters, the a-counter that counts the number of as, the b-counter, and the c-counter, initializes the a and b counters to 0, the c-counter to 1, and becomes ready. When an agent in state a interacts with an agent in state c, a becomes $\bar{a}$ to indicate that the agent is now sleeping, and c does the following (in fact, we assume that c goes to a special state c_a in which it knows that it has seen an a, and that all the following are done internally, after the interaction; finally the agent restores its state to c and becomes again ready for interaction): it increases its a-counter by one (in binary), multiplies its a- and b-counters, which can be done in binary in logarithmic space (binary multiplication is in *LOGSPACE*), compares the result with the c-counter, copies the result of the comparison to its output tape, that is, 1 if they are equal and 0 otherwise, and finally it copies the comparison result and its three counters to the message tape and becomes ready for interaction. Similar things happen when a b meets a c (interchange the roles of a and b in the above discussion). When a c meets a c, the responder becomes $\bar{c}$ and copies to its output tape the output bit contained in the initiator's message. The initiator remains to c, adds the a-counter contained in the responder's message to its a-counter, the b- and c-counters of the message to its b- and c-counters, respectively, multiplies again the updated a- and b-counters, compares the result to its updated c-counter, stores the comparison result to its output and message tapes, copies its counters to its message tape, and becomes ready again. When a $\bar{a}$, $\bar{b}$, or $\bar{c}$ meets a c they only copy to their output tape the output bit contained in c's message and become ready again (e.g., $\bar{a}$ remains $\bar{a}$), while c does nothing.

Note that the number of cs is at most n which means that the c-counter will become at most $\lceil \log n \rceil$ bits long, and the same holds for the a- and b-counters, so there is enough room in the tape of an agent to store them.

Given a fair execution, eventually only one agent in state c will remain, its a-counter will contain the total number of as, its b-counter the total number of bs, and its c-counter the total number of cs. By executing the multiplication of the a- and b-counters and comparing the result to its c-counter it will correctly determine whether $(N_c = N_a \cdot N_b)$ holds and it will store the correct result (0 or 1) to its output and message tapes. At that point all other agents will be in one of the states $\bar{a}$, $\bar{b}$, and $\bar{c}$. All these, again due to fairness, will eventually meet the unique agent in state

c and copy its correct output bit (which they will find in the message they get from c) to their output tapes. Thus, eventually all agents will output the correct value of the predicate. □

Exercise 14 Prove that the basic PALOMA model is strictly stronger than the basic population protocol model, without exploiting the predicate $(N_c = N_a \cdot N_b)$.
Hint: Find another non semilinear predicate that is (stably) computable by the basic PALOMA model. Do not forget to show first that the basic PALOMA model is at least as strong as the basic population protocol model.

Definition 15 Let *PLM* denote the class of all symmetric predicates p that are stably computable by the basic PALOMA model.

Then, one can prove the following exact characterization for PLM [14]. Unfortunately, this proof is also quite involved and due to space restrictions we skip it.

Theorem 7 *PLM is equal to the class of all symmetric predicates in* $NSPACE(n \log n)$.

5.7 Algorithmic Verification of Population Protocols

In order to apply our protocols to real-critical application scenarios, some form of computer-aided *verification* is necessary. Even if a protocol is followed by a formal proof of correctness it would be safer to verify its code before loading it to the real sensor nodes.

It seems that the easiest (but not easy) place to start the investigation of verification is the basic population protocol model. In this model we can exploit the fact that symmetry allows a configuration to be safely represented as a $|Q|$-vector of non-negative integers. This section, based on [13], will reveal the inherent hardness of algorithmic verification of basic population protocols but will also present a promising algorithmic solution.

Section 5.7.1 provides all necessary definitions. Section 5.7.2 deals with the hardness of algorithmic verification of basic population protocols; the general problem and many of its special cases are proved to be hard. Section 5.7.3 studies an efficiently solvable, but though almost trivial, special case. Finally, Sect. 5.7.4 presents some non complete and one complete algorithmic solution.

5.7.1 Necessary Definitions

5.7.1.1 Population Protocols

We begin by revising all relevant definitions concerning the basic population protocol model, most of which are now presented in an alternative manner, because throughout this section we will exploit the fact that symmetry allows a configuration to be represented as a $|Q|$-vector of nonnegative integers, and there is no need now to use a function for this purpose.

In this section, the transition function δ is also treated as a relation $\Delta \subseteq Q^4$, defined as $(q_i, q_j, q_l, q_t) \in \Delta$ iff $\delta(q_i, q_j) = (q_l, q_t)$. We assume that the communication graph is a complete digraph, without self-loops and multiple edges (that is, we deal with the basic model). We denote by G^k the complete communication graph of k nodes.

Let now $k \equiv |V|$ denote the *population size*. An *input assignment* x is a mapping from $V = [k]$ to X (where $[l]$, for $l \in \mathbb{Z}_{\geq 1}$, denotes the set $\{1, \ldots, l\}$), assigning an input symbol to each agent of the population. As already mentioned in Sect. 5.2.2, since the communication graph is complete, due to symmetry, we can, equivalently, think of an input assignment as a $|X|$-vector of integers $x = (x_i)_{i \in [|X|]}$, where, for all i, x_i is nonnegative and equal to the number of agents that receive the symbol $\sigma_i \in X$, assuming an ordering on the input symbols. We denote by $\mathcal{X}$ the set of all possible input assignments. Note that for all $x \in \mathcal{X}$ it holds that $\sum_{i=1}^{|X|} x_i = k$.

A state $q \in Q$ is called *initial* if $I(\sigma) = q$ for some $\sigma \in X$. A *configuration* c is a mapping from $[k]$ to Q, so, again, it is a $|Q|$-vector of non-negative integers $c = (c_i)_{i \in [|Q|]}$ such that $\sum_{i=1}^{|Q|} c_i = k$ holds. Each input assignment corresponds to an *initial configuration* which is indicated by the input function I. In particular, input assignment x corresponds to the initial configuration $c(x) = (c_i(x))_{i \in [|Q|]}$, where $c_i(x)$ is equal to the number of agents that get some input symbols σ_j for which $I(\sigma_j) = q_i$ (q_i is the ith state in Q if we assume the existence of an ordering on the set of states Q). More formally, $c_i(x) = \sum_{j: I(\sigma_j) = q_i} x_j$ for all $i \in [|Q|]$. By extending I to a mapping from input assignments to configurations we can write $I(x) = c$ to denote that c is the initial configuration corresponding to input assignment x. Let $\mathcal{C} = \left\{ (c_i)_{i \in [|Q|]} \mid c_i \in \mathbb{Z}^+ \text{ and } \sum_{i=1}^{|Q|} c_i = k \right\}$ denote the set of all possible configurations given the population protocol $\mathcal{A}$ and G^k. Moreover, let $C_I = \{c \in \mathcal{C} \mid I(x) = c \text{ for some } x \in \mathcal{X}\}$ denote the set of all possible initial configurations. Any $r \in \Delta$ has four components which are elements from Q and we denote by r_i, where $i \in [4]$, the ith component (i.e., state) of r. $r \in Q^4$ belongs to Δ iff $\delta(r_1, r_2) = (r_3, r_4)$. We say that a configuration c can go in one step to c' via transition $r \in \Delta$, and write $c \xrightarrow{r} c'$, if

- $c_i \geq r_{1,2}(i)$, for all $i \in [|Q|]$ for which $q_i \in \{r_1, r_2\}$,
- $c'_i = c_i - r_{1,2}(i) + r_{3,4}(i)$, for all $i \in [|Q|]$ for which $q_i \in \{r_1, r_2, r_3, r_4\}$, and
- $c'_j = c_j$, for all $j \in [|Q|]$ for which $q_j \in Q - \{r_1, r_2, r_3, r_4\}$,

where $r_{l,t}(i)$ denotes the number of times state q_i appears in (r_l, r_t). Moreover, we say that a configuration c can go in one step to a configuration c', and write $c \rightarrow c'$ if $c \xrightarrow{r} c'$ for some $r \in \Delta$. We say that a configuration c' is reachable from a configuration c, denoted $c \xrightarrow{*} c'$ if there is a sequence of configurations $c = c^0, c^1, \ldots, c^t = c'$, such that $c^i \rightarrow c^{i+1}$ for all i, $0 \leq i < t$, where c^i denotes here the $(i+1)$th configuration of an execution (and not the ith component of configuration c which is denoted c_i). An *execution* is a finite or infinite sequence of configurations $c^0, c^1, \ldots$, so that $c^i \to c^{i+1}$. An execution is *fair* if for all configurations c, c' such that $c \to c'$, if c appears infinitely often then so does c'. A *computation* is an infinite fair execution. A predicate p is said to be *stably computable* by a PP $\mathcal{A}$ if, for any input assignment x, any computation of $\mathcal{A}$ contains an *output-stable configuration* in which all agents output $p(x)$. A configuration c is called *output-stable* if $O(c) = O(c')$, for all c' reachable from c (where O, here, is an extended version of the output function from configurations to output assignments in Y^k). We denote by $C_F = \{c \in \mathcal{C} \mid c \to c' \Rightarrow c' = c\}$ the set of all *final configurations*. We can further extend the output function O to a mapping from configurations to $\{-1, 0, 1\}$, defined as

$$O(c) = \begin{cases} 0, & \text{if } O(c(u)) = 0, \text{ for all } u \in V \\ 1, & \text{if } O(c(u)) = 1, \text{ for all } u \in V \\ -1, & \text{if } \exists u, \upsilon \in V \text{ s.t. } O(c(u)) \neq O(c(\upsilon)). \end{cases}$$

It is known [2, 6] that a predicate is stably computable by the PP model iff it can be defined as a first-order logical formula in *Presburger arithmetic*. Let ϕ be such a formula. There exists some PP that stably computes ϕ, thus ϕ constitutes, in fact, the specifications of that protocol. For example, consider the formula $\phi = (N_a \geq 2N_b)$. ϕ partitions the set of all input assignments, $\mathcal{X}$, to those input assignments that satisfy the predicate (that is, the number of as assigned is at least two times the number of bs assigned) and to those that do not. Moreover, ϕ can be further extended to a mapping from C_I to $\{-1, 0, 1\}$. In this case, ϕ is defined as

$$\phi(c) = \begin{cases} 0, & \text{if } \phi(x) = 0, \text{ for all } x \in I^{-1}(c) \\ 1, & \text{if } \phi(x) = 1, \text{ for all } x \in I^{-1}(c) \\ -1, & \text{if } \exists x, x' \in I^{-1}(c) \text{ s.t. } \phi(x) \neq \phi(x'), \end{cases}$$

where $I^{-1}(c)$ denotes the set of all $x \in \mathcal{X}$ for which $I(x) = c$ holds (the *preimage* of c).

We now define the *transition graph*, which is similar to that defined in [2], except for the fact that it here contains only those configurations that are reachable from some initial configuration in C_I. Specifically, given a population protocol $\mathcal{A}$ and an integer $k \geq 2$ we can define the transition graph of the pair $(\mathcal{A}, k)$ as $G_{\mathcal{A},k} = (C_r, E_r)$, where the node set $C_r = C_I \cup \{c \in \mathcal{C} \mid c' \xrightarrow{*} c \text{ for some } c' \in C_I\}$ of G_r (we use G_r as a shorthand of $G_{\mathcal{A},k}$) is the subset of $\mathcal{C}$ containing all initial configurations and all configurations that are reachable from some initial one, and the edge (or arc) set $E_r = \{(c, c') \mid c, c' \in C_r \text{ and } c \to c'\}$ of G_r contains a directed edge (c, c') for any two (not necessarily distinct) configurations c and c' of C_r for which it holds that c can go in one step to c'. Note that G_r is a directed (weakly) connected graph with possible self-loops. It was shown in [2] that given a computation Ξ, the configurations that appear infinitely often in Ξ form a *final strongly connected component* of G_r. We denote by S the collection of all strongly connected components of G_r. Note that each $B \in S$ is simply a set of configurations.

Moreover, given $B, B' \in S$ we say that the B can go in one step to B', and write $B \to B'$, if $c \to c'$ for $c \in B$ and $c' \in B'$. $B \xrightarrow{*} B'$ is defined as in the case of configurations. We denote by $I_S = \{B \in S \mid \text{such that } B \cap C_I \neq \emptyset\}$ those components that contain at least one initial configuration, and by $F_S = \{B \in S \mid \text{such that } B \to B' \Rightarrow B' = B\}$ the final ones. We can now extend ϕ to a mapping from I_S to $\{-1, 0, 1\}$ defined as

$$\phi(B) = \begin{cases} 0, & \text{if } \phi(c) = 0, \text{ for all } c \in B \cap C_I \\ 1, & \text{if } \phi(c) = 1, \text{ for all } c \in B \cap C_I \\ -1, & \text{if } \exists c, c' \in B \cap C_I \text{ s.t. } \phi(c) \neq \phi(c'), \end{cases}$$

and O to a mapping from F_S to $\{-1, 0, 1\}$ defined as

$$O(B) = \begin{cases} 0, & \text{if } O(c) = 0, \text{ for all } c \in B \\ 1, & \text{if } O(c) = 1, \text{ for all } c \in B \\ -1, & \text{otherwise.} \end{cases}$$

5.7.1.2 Problems' Definitions

We begin by defining the most interesting and natural version of the problem of algorithmically verifying basic population protocols. We call it *GBPVER* ("G" standing for "General', "B" for "Basic", and "P" for "Predicate") and its complement $\overline{\textit{GBPVER}}$ is defined as follows:

Problem 2 ($\overline{\textit{GBPVER}}$) Given a population protocol $\mathcal{A}$ for the basic model whose output alphabet $Y_{\mathcal{A}}$ is binary (i.e., $Y_{\mathcal{A}} = \{0, 1\}$) and a first-order logical formula ϕ in Presburger arithmetic representing the specifications of $\mathcal{A}$, determine whether there exists some integer $k \geq 2$ and some legal input assignment x for the complete communication graph of k nodes, G^k, for which not all computations of $\mathcal{A}$ on G^k beginning from the initial configuration corresponding to x stabilize to the correct output w.r.t. ϕ.

A special case of *GBPVER* is *BPVER* (its nongeneral version as revealed by the missing "G"), and is defined as follows.

Problem 3 (*BPVER)* Given a population protocol $\mathcal{A}$ for the basic model whose output alphabet $Y_{\mathcal{A}}$ is binary (i.e., $Y_{\mathcal{A}} = \{0, 1\}$), a first-order logical formula ϕ in Presburger arithmetic representing the specifications of $\mathcal{A}$, and an integer $k \geq 2$ (in binary) determine whether $\mathcal{A}$ conforms to its specifications on G^k.

"Conforms to ϕ" here means that for any legal input assignment x, which is a $|X_{\mathcal{A}}|$-vector with nonnegative integer entries that sum up to k, and any computation beginning from the initial configuration corresponding to x on G^k, the population stabilizes to a configuration in which all agents output the value $\phi(x) \in \{0, 1\}$ (that is, it is equivalent to "stably computes", but we now view it from the verification perspective). On the other hand, "does not conform" means that there is at least one computation of $\mathcal{A}$ on G^k which is unstable or the stable output does not agree with $\phi(x)$—i.e., not all agents output the value $\phi(x)$.

Problem 4 (*BBPVER*) *BBPVER* (the additional "B" is from "Binary input alphabet") is *BPVER* with $\mathcal{A}$'s input alphabet restricted to {0, 1}.

5.7.2 NP-Hardness Results

5.7.2.1 *BP* Verification

We first show that *BPVER* is a hard computational problem.

Theorem 8 *BPVER is coNP-hard.*

Proof We shall present a polynomial-time reduction from $HAMPATH = \{< D, s, t > \mid D$ is a directed graph with a Hamiltonian path from s to t } to $\overline{BPVER}$. In other words, we will present a procedure that given an instance $< D, s, t >$ of *HAMPATH* returns in polynomial time an instance $< \mathcal{A}, \phi, k >$ of $\overline{BPVER}$, such that $< D, s, t > \in HAMPATH$ iff $< \mathcal{A}, \phi, k > \in \overline{BPVER}$. If there is a hamiltonian path from s to t in D we will return a population protocol $\mathcal{A}$ that for some computation on the complete graph of k nodes fails to conform to its specification ϕ, and if there is no such path all computations will conform to ϕ.

We assume that all nodes in $V(D) - \{s, t\}$ are named $q_1, \ldots, q_{n-2}$, where n denotes the number of nodes of D (be careful: n does not denote the size of the population, but the number of nodes of the graph D in *HAMPATH*'s instance). We now construct the protocol $\mathcal{A} = (X, Y, Q, I, O, \delta)$. The output alphabet Y is {0, 1} by definition. The input alphabet X is $E(D) - (\{(\cdot, s)\} \cup \{t, \cdot\})$, that is, consists of all edges of D except for those leading into s and those going out of t. The set of states Q is equal to $X \cup T \cup \{r\}$, where $T = \{(s, q_i, q_j, l) \mid 1 \leq i, j \leq n-2$ and $1 \leq l \leq n-1\}$ and its usefulness will be explained later. r can be thought of as being the "reject" state, since we will define it to be the only state giving the output value 0. Notice that $|Q| = \mathcal{O}(n^3)$. The input function $I : X \to Q$ is defined as $I(x) = x$, for all $x \in X$, and for the output function $O : Q \to \{0, 1\}$ we have $O(r) = 0$ and $O(q) = 1$ for all $q \in Q - \{r\}$. That is, all input symbols are mapped to themselves, while all states are mapped to the output value 1, except for r which is the only state giving 0 as output. Thinking of the transition function δ as a transition matrix Δ it is easy to see that Δ is a $|Q| \times |Q|$ matrix whose entries are elements from $Q \times Q$. Each entry $\Delta_{q,q'}$ corresponds to the rhs of a rule $(q, q') \to (z, z')$ in δ. Clearly, Δ consists of $\mathcal{O}(n^6)$ entries, which is again polynomial in n.

We shall postpone for a while the definition of Δ to first define the remaining parameters ϕ and k of $\overline{BPVER}$'s instance. We define formula ϕ to be a trivial first-order Presburger arithmetic logical formula that is always false. For example, in the natural nontrivial case where $X \neq \emptyset$ (that is, D has at least one edge that is not leading into s and not going out of t) we can pick any $x \in X$ and set $\phi = (N_x < 0)$ which, for N_x denoting the number of x's appearing in the input assignment, is obviously always false. It is useful to notice that the only configuration that gives the correct output w.r.t. ϕ is the one in which all agents are in state r. ϕ being always false means that in a correct protocol all computations must stabilize to the all-zero output, and r is the only state giving output 0. On the other hand for $\mathcal{A}$ not to be correct w.r.t. ϕ it suffices to show that there exists some computation in which r cannot appear. Moreover, we set k equal to $n-1$, that is, the communication graph on which $\mathcal{A}$'s correctness has to be checked by the verifier is the complete digraph of $n-1$ nodes (or, equivalently, agents).

To complete the reduction, it remains to construct the transition function δ:

- $(r,\cdot) \to (r,r)$ and $(\cdot,r) \to (r,r)$ (so r is a propagating state, meaning that once it appears it eventually becomes the state of every agent in the population)
- $((q_i,q_j),(q_i,q_j)) \to (r,r)$ (if two agents get the same edge of D then the protocol rejects)
- $((q_i,q_j),(q_i,q_l)) \to (r,r)$ (if two agents get edges of D with adjacent tails then the protocol rejects)
- $((q_j,q_i),(q_l,q_i)) \to (r,r)$ (if two agents get edges of D with adjacent heads then the protocol rejects—it also holds if one of q_j and q_l is s)
- $((q_i,t),(q_j,t) \to (r,r)$ (the latter also holds for the sink t)
- $((s,\ldots),(s,\ldots)) \to (r,r)$ (if two agents have both s as the first component of their states then the protocol rejects)
- $((s,q_i),(q_i,q_j)) \to ((s,q_i,q_j,2),(q_i,q_j))$ (when s meets an agent υ that contains a successor edge it keeps q_j to remember the head of υ's successor edge and releases a-counter set to 2 - it counts the number of edges encountered so far on the path trying to reach t from s)
- $((s,q_i,q_j,i),(q_j,q_l)) \to ((s,q_i,q_l,i+1),(q_j,q_l))$, for $i < n-2$
- $((s,q_i,q_j,i),(q_j,t)) \to (r,r)$, for $i < n-2$ (the protocol rejects if s is connected to t through a directed path with less than $n-1$ edges)
- All the transitions not appearing above are identity rules (i.e., they do nothing)

Now we prove that the above, obviously polynomial-time, construction is in fact the desired reduction. If D contains some hamiltonian path from s to t, then the $n-1$ edges of that path form a possible input assignment to protocol $\mathcal{A}$ (since its input symbols are the edges and the population consists of $n-1$ agents). When $\mathcal{A}$ gets that input it cannot reject (r cannot appear) for the following reasons:

- no two agents get the same edge of D
- no two agents get edges of D with adjacent tails
- no two agents get edges of D with adjacent heads
- only one $(s,\ldots)$ exists
- s cannot count less than $n-1$ edges from itself to t

So, when $\mathcal{A}$ gets the input alluded to above, it cannot reach state r; thus, it cannot reject, which implies that $\mathcal{A}$ for that input always stabilizes to the wrong output w.r.t. ϕ (which always requires the "reject" output) when runs on the G^{n-1}. So, in this case $<\mathcal{A},\phi,k>$ consists of a protocol $\mathcal{A}$ that, when runs on G^k, where $k=n-1$, for a specific input it does not conform to its specifications as described by ϕ, so clearly it belongs to $\overline{BPVER}$.

For the other direction, if $<\mathcal{A},\phi,k>\in \overline{BPVER}$ then obviously there exists some computation of $\mathcal{A}$ on the complete graph of $k=n-1$ nodes in which r does not appear at all (if it had appeared once then, due to fairness, the population would have stabilized to the all-r configuration, resulting to a computation conforming to ϕ). It is helpful to keep in mind that most arguments here hold because of the fairness condition. Since r cannot appear, every agent (of the $n-1$ in total) must have been assigned a different edge of D. Moreover, no two of them contain edges with common tails or common heads in D. Note that there is only one agent with state $(s,\ldots)$ because if there were two of them they would have rejected when interacted with each other, and if no $(s,\ldots)$ appeared then two agents would have edges with common tails because there are $n-1$ edges for $n-2$ candidate initiating points (we have not allowed t to be an initiating point) and the pigeonhole principle applies (and by symmetric arguments only one with state $(\ldots,t)$). So, in the induced graph formed by the edges that have been assigned to the agents, s has outdegree 1 and indegree 0, t has indegree 1 and outdegree 0, and all remaining nodes have indegree at most 1 and outdegree at most 1. This implies that all nodes except for s and t must have indegree equal to 1 and outdegree equal to 1. If, for example, some node had indegree 0, then the total indegree could not have been $n-1$ because $n-3$ other nodes have indegree at most 1, t has indegree 1, and s has 0 (the same holds for outdegrees). Additionally, there is some path initiating from s and ending to t. This holds because the path initiating from s (s has outdegree 1) cannot fold upon itself (this would result in a node with indegree greater than 1) and cannot end to any other node different from t because this would result to some node other than t with outdegree equal to 0. Finally, that path has at least $n-1$ edges (in fact, precisely $n-1$ edges), since if it had less the protocol would have rejected. Thus, it must be clear after the above discussion that in this case there must have been a hamiltonian path from s to t in D, implying that $< D, s, t > \in$ *HAMPATH*. □

Note that in the above reduction the communication graph has only $\mathcal{O}(n)$ nodes while the protocol has size $\mathcal{O}(n^6)$. Although this is not the usual case, it is not forbidden because this concerns only the correctness of the protocol on this specific complete graph. The protocol remains independent of the population size; it will still count up to $n-1$ while the population can have arbitrarily large size (another way to think of this is that in the protocol description the population size is not a parameter). The protocol may be wrong or correct for other combinations of specifications and communication graphs but we do not care here. However, it is worth considering the following question: "Can we also prove that the special case of *BPVER* in which the protocol has always size less than the size of the communication graph (which is the natural scenario) is coNP-hard?" Unfortunately, the answer to this question is that we do not know yet.

5.7.2.2 *BBP* Verification

We now deal with the hardness of *BBPVER* (here, additionally, we have a binary input alphabet).

Theorem 9 *BBPVER is coNP-hard.*

Proof The reduction is again from *HAMPATH* to $\overline{\mathit{BBPVER}}$. Let again $< D, s, t >$ be the instance of *HAMPATH*. X is equal to $\{0, 1\}$ as is required by definition and so is Y. Q is again equal to $(E(D) - (\{(\cdot, s)\} \cup \{t, \cdot\})) \cup T \cup \{q_0, t', r\}$, where $T = \{(s, q_i, q_j, l) \mid 1 \leq i, j \leq n-2$ and $1 \leq l \leq n-1\}$. The input function I is defined as $I(0) = (s, f^+(s))$, where $f^+(s)$ is the first (smallest) out-neighbor of s according to the lexicographic order of $V(D)$ (if some node u has no neighbors we assume that $f^+(u) = u$), and $I(1) = q_0$ (recall that the names that we use for nodes are $s, t, q_1, \ldots, q_{n-2}$ so q_0 is just a special initial state). The output function O again maps all states in $Q - \{r\}$ to 1 and r to 0. ϕ is an always false predicate and k is set to $n-1$, where $n = |V(G)|$.

We now define the transition function.

- $(r, \cdot) \rightarrow (r, r)$ and $(\cdot, r) \rightarrow (r, r)$ (so r is a propagating state, meaning that once it appears it eventually becomes the state of every agent in the population)
- $((q_i, q_j), (q_i, q_j)) \rightarrow (r, r)$ (if two agents have obtained the same edge of D then the protocol rejects)
- $((q_i, q_j), (q_i, q_l)) \rightarrow (r, r)$ (if two agents have obtained edges of D with adjacent tails then the protocol rejects)
- $((q_j, q_i), (q_l, q_i)) \rightarrow (r, r)$ (if two agents have obtained edges of D with adjacent heads then the protocol rejects - it also holds if one of q_j and q_l is s)
- $(q_0, q_0) \rightarrow (r, r)$
- $((s, \ldots), (s, \ldots)) \rightarrow (r, r)$
- $(\cdot, t), (\cdot, t) \rightarrow (r, r)$
- $((s, q_i), q_0) \rightarrow ((s, q_i), (f^-(t), t))$ (where $f^-(t)$ denotes the first (smallest) in-neighbor of t according to the lexicographic order; we can w.l.o.g. assume that t has at least one incoming edge)
- $((s, q_i), (q_j, t)) \rightarrow ((s, h_s^+(q_i)), (q_j, t))$ (where $h_s^+(q_i)$ denotes the lexicographically smallest out-neighbor of s that is lexicographically greater than q_i (that is, the next one); note that the lexicographically greatest is matched to the lexicographically smallest in a cyclic fashion)
- $((q_j, t), (s, q_i)) \rightarrow ((h_t^-(q_j), t), (s, q_i))$ (where $h_t^-(q_j)$ denotes the lexicographically smallest in-neighbor of t that is lexicographically greater than q_j)
- $((q_i, q_j), (q_l, t)) \rightarrow ((), ())$
- $(q_0, (s, q_i)) \rightarrow ((q_i, f^+(q_i)), (s, q_i))$, if $f^+(q_i) \neq q_i$, and (r, r), otherwise (if $f^+(q_i) = q_i$ then q_i has no outgoing neighbors and the protocol rejects; f^+ does not take into account the edges leading into s and t)
- $((q_i, q_j), (q_l, t)) \rightarrow ((q_i, h_{q_i}^+(q_j)), (q_l, t))$
- $((q_l, t), (q_i, q_j)) \rightarrow ((q_l, t), (q_j, f^+(q_j)))$, if $f^+(q_j) \neq q_j$, and (r, r), otherwise
- $((s, q_i), (q_i, q_j)) \rightarrow ((s, q_i, q_j, 2), (q_i, q_j))$ (when s meets an agent υ that contains a successor edge it keeps q_j to remember the head of υ's successor edge and releases a counter set to 2—it counts the number of edges encountered so far on the path trying to reach t from s)
- $((s, q_i, q_j, i), (q_j, q_l)) \rightarrow ((s, q_i, q_l, i+1), (q_j, q_l))$, for $i < n-2$

- $((s, q_i, q_j, i), (q_j, t)) \to (r, r)$, for $i < n-2$ (the protocol rejects if s is connected to t through a directed path with less than $n-1$ edges)
- $((s, q_i, q_j, n-2), (q_j, t)) \to ((s, q_i, q_j, n-1), t')$
- t' and q_0 reject any t' and $(\cdot, t)$ that they encounter
- All the transitions not appearing above are identity rules (i.e., they do nothing)

Given a hamiltonian path $s, u_1, \ldots, u_{n-2}, t$ of D we present an erroneous computation of $\mathcal{A}$ on the complete digraph of $k = n-1$ nodes w.r.t. ϕ (that is, a computation in which state r does not appear). A possible input assignment is the 2-vector $(1, n-2)$ in which one agent gets input 0 and $(n-2)$ agents get input 1. So, the initial configuration corresponding to this input has one agent in $(s, f^+(s))$ and all the other agents in q_0. The agent in $(s, f^+(s))$ now interacts as the initiator with some agent and $(f^-(t), t)$ appears. Now we have one agent in $(s, f^+(s))$, one in $(f^-(t), t)$, and all the remaining in q_0. If $f^+(s)$ is not equal to u_1 (the second node in the hamiltonian path) we assume that $(s, f^+(s))$ is the initiator of as many interactions with $(f^-(t), t)$ as needed to make $(s, f^+(s))$ go to (s, u_1). Similarly, with $(f^-(t), t)$ being the initiator we make it interact a sufficient number of times with (s, u_1) so that it becomes (u_{n-2}, t). Now one agent contains (s, u_1), which is the first edge of the hamiltonian path, one agent contains (u_{n-2}, t) which is the last edge of the hamiltonian path, and all remaining agents are in q_0. Now interaction $(q_0, (s, u_1))$ takes place and the result is $((u_1, f^+(u_1)), (s, u_1))$, where $f^+(u_1)$ is the lexicographically first out-neighbor of u_1, which is possibly not u_2. If it is not, then we let the agent which is in $(u_1, f^+(u_1))$ repeatedly interact as the initiator with (u_{n-2}, t), until its state becomes (u_1, u_2) (e.g., during the first interaction $(u_1, f^+(u_1))$ becomes $(u_1, h^+_{u_1}(f^+(u_1)))$, where $h^+_{u_1}(u)$ denotes the out-neighbor of u_1 lexicographically following u). As soon as this happens, (s, u_1) interacts with another agent in q_0 which again updates its state to $(u_1, f^+(u_1))$. Again $(u_1, f^+(u_1))$ interacts as the initiator with (u_{n-2}, t) as many times as needed to make its state (u_1, u_2) and then it interacts once as the responder with (u_{n-2}, t) to change its state to $(u_2, f^+(u_2))$. Even if $f^+(u_2)$ does not happen to be u_3 we can force it to be by subsequent interactions with (u_{n-2}, t) (with the latter now being the responder). In this manner we can easily make each agent in the population contain a different edge of the hamiltonian path. Moreover, notice that we have not allowed any interaction that leads to failure (i.e., that makes state r appear) happen. Now (s, u_1) meets (u_1, u_2) and the former becomes $(s, u_1, u_2, 2)$. Then it meets (u_2, u_3) and becomes $(s, u_1, u_3, 3)$, and so on, and, finally, when it has become $(s, u_1, u_{n-2}, n-2)$ it meets (u_{n-2}, t) and after that interaction the former becomes $(s, u_1, u_{n-2}, n-1)$ and the latter t'. It is easy now to observe that from this point on there is no possible interaction that could make r appear and thus we have just presented an erroneous computation (all agents forever output the value 1, but ϕ requires that any computation stabilizes to the all-zero output). The convincing argument that it is a computation (i.e., a fair execution) is that we keep the execution unfair only for a finite number of steps, which does not violate the fairness condition.

For the inverse, let us assume that there exists some computation of $\mathcal{A}$ on the complete digraph of $k = n-1$ nodes in which r never appears. Clearly, only one $(s, \ldots)$ ever appears (if there were two of them they would eventually meet and reject, because once s appears as the first component of some state it cannot be eliminated, and if there was none the population would solely consist of q_0, which would eventually meet and reject). Similarly, only one t ever appears (since, once they appear, even if they ever become t', they cannot be eliminated and will eventually meet each other and reject). Note also that after a finite number of steps all agents must have obtained some edge (if some agent remains forever in q_0 then it eventually meets $(\cdot, t)$ or t' and rejects). Moreover, t' must have appeared for the following reason: if not, then $(\cdot, t)$ would forever change the agents' edges, so due to fairness two agents would, in a finite number of steps, obtain the same edge, interact with each other, and reject. But since the protocol cannot reject (in the computation under consideration), s must have counted to $n-2$ before meeting t, and by repeating some of the arguments used in the proof of Theorem 8 one can again show that any node in the induced graph (constructed by the edges contained in the agents) has indegree equal to 1 and outdegree equal to 1, which implies that there must exist some hamiltonian path from s to t in D. Clearly, this completes the proof. □

Notice now that Theorem 9 constitutes an immediate alternative proof for Theorem 8. To see this, observe that any protocol with binary input is also a protocol with general input. Thus, in the case where $\mathcal{A}$ has a binary input alphabet, $< \mathcal{A}, \phi, k > \in \overline{BBPVER}$ is a sufficient and necessary condition for $< \mathcal{A}, \phi, k > \in \overline{BPVER}$, which establishes $\overline{BBPVER} \leq_p \overline{BPVER}$.

5.7.2.3 *BPVER′* and *BBPVER′*

Let us denote by $BPVER'$ the special case of $BPVER$ in which the protocol size is at least the size k of the communication graph, and similarly for $BBPVER'$. Clearly, the proofs of Theorems 8 and 9 establish that both problems are coNP-hard.

5.7.2.4 *GBP* Verification

We now study the hardness of $\overline{GBPVER}$.

Theorem 10 *GBPVER is coNP-hard.*

Proof We will prove the statement by presenting a polynomial-time reduction from $\overline{BPVER'}$ to $\overline{GBPVER}$. Every time that we get an instance $< \mathcal{A}, \phi, k >$ of $\overline{BPVER'}$ (where $\mathcal{A}$ is a population protocol for which $| < A > | \geq k$ holds), if $\mathcal{A}$ has a computation on G^k that does not stabilize to the correct output w.r.t. ϕ then we will return a population protocol $\mathcal{A}'$ and a formula ϕ' such that there exists some k' for which $\mathcal{A}'$ has a computation on $G^{k'}$ that does not stabilize to the correct output w.r.t. ϕ'. On the other hand, if $\mathcal{A}$ has no such erroneous computation on G^k, $\mathcal{A}'$ will also have no erroneous computation (w.r.t. ϕ') for any complete communication graph (of any size greater than or equal to 2). Moreover, we will achieve that in polynomial time.

Keep in mind that the input to the machine computing the reduction is $< \mathcal{A}, \phi, k >$. Let $X_{\mathcal{A}}$ be the input alphabet of $\mathcal{A}$. Clearly, $\phi'' = \neg(\sum_{x \in X_{\mathcal{A}}} N_x = k)$ is a semilinear predicate if k is treated as a constant (N_x denotes the number of agents with input x). Thus, there exists a population protocol $\mathcal{A}''$ for the basic model that stably computes ϕ''. The population protocol $\mathcal{A}''$ can be constructed efficiently. Its input alphabet $X_{\mathcal{A}''}$ is equal to $X_{\mathcal{A}}$. The construction of the protocol can be found in [2] (in fact they present there a more general protocol for any linear combination of variables corresponding to a semilinear predicate). When the number of nodes of the communication graph is equal to k, $\mathcal{A}''$ always stabilizes to the all-zero output (all agents output the value 0) and when it is not equal to k, then $\mathcal{A}''$ always stabilizes to the all-one output.

We want to construct an instance $< \mathcal{A}', \phi' >$ of $\overline{GBPVER}$. We set $\phi' = \phi \vee \phi''$. Moreover, $\mathcal{A}'$ is constructed to be the composition of $\mathcal{A}$ and $\mathcal{A}''$. Obviously, $Q_{\mathcal{A}'} = Q_{\mathcal{A}} \times Q_{\mathcal{A}''}$. We define its output to be the union of its components' outputs, that is, $O(q_{\mathcal{A}}, q_{\mathcal{A}''}) = 1$ iff at least one of $O(q_{\mathcal{A}})$ and $O(q_{\mathcal{A}''})$ is equal to 1. It is easy to see that the above reduction can be computed in polynomial time.

We first prove that if $< \mathcal{A}, \phi, k > \in \overline{BPVER'}$ then $< \mathcal{A}', \phi' > \in \overline{GBPVER}$. When $\mathcal{A}'$ runs on the complete graph of k nodes, the components of its states corresponding to $\mathcal{A}''$ stabilize to the all-zero output, independently of the initial configuration. Clearly, $\mathcal{A}'$ in this case outputs whatever $\mathcal{A}$ outputs. Moreover, for this communication graph, ϕ' is true iff ϕ is true (because $\phi'' = \neg(\sum_{x \in X_{\mathcal{A}}} N_x = k)$ is false, and $\phi' = \phi \vee \phi''$). But there exists some input for which $\mathcal{A}$ does not give the correct output with respect to ϕ (e.g., ϕ is true for some input but $\mathcal{A}$ for some computation does not stabilize to the all-one output). Since ϕ' expects the same output as ϕ and $\mathcal{A}'$ gives the same output as $\mathcal{A}$ we conclude that there exists some erroneous computation of $\mathcal{A}'$ w.r.t. ϕ', and the first direction has been proven.

Now, for the other direction, assume that $< \mathcal{A}', \phi' > \in \overline{GBPVER}$. For any communication graph having a number of nodes not equal to k, ϕ' is true and $\mathcal{A}'$ always stabilizes to the all-one output because of the $\mathcal{A}''$ component. This means that the erroneous computation of $\mathcal{A}'$ happens on the G^k. But for that graph, ϕ'' is always false and $\mathcal{A}''$ always stabilizes its corresponding component to the all-zero output. Now ϕ' is true iff ϕ is true and $\mathcal{A}'$ outputs whatever $\mathcal{A}$ outputs. But there exists some input and a computation for which $\mathcal{A}'$ does not stabilize to a configuration in which all agents give the output value that ϕ' requires which implies that $\mathcal{A}$ does not stabilize to a configuration in which all agents give the output value required by ϕ. Since the latter holds for G^k, the theorem follows. □

5.7.2.5 *BBPI* Verification

To show the inherent difficulty of the population protocol verification problem we consider an even simpler special case, namely, the $BBPIVER$ problem ("I" standing for "Input", because an input assignment is additionally provided as part of the algorithm's input) that is defined as follows:

Problem 5 (*BBPIVER*) Given a population protocol $\mathcal{A}$ for the basic model whose input and output alphabets are binary (i.e., $X_{\mathcal{A}} = Y_{\mathcal{A}} = \{0, 1\}$), a two-variable first-order logical formula ϕ in Presburger arithmetic representing the specifications of $\mathcal{A}$, and an input (assignment) $x = (x_0, x_1)$, where x_0 and x_1 are nonnegative integers, determine whether $\mathcal{A}$ conforms to its specifications for the complete digraph of $k = x_0 + x_1$ nodes whenever its computation begins from the initial configuration corresponding to x.

Let *BBPIVER*$'$ denote the special case of *BBPIVER* in which $| < \mathcal{A} > | \geq k$.

Theorem 11 *BBPIVER*$'$ *and BBPIVER are coNP-hard.*

Proof The reduction is from *HAMPATH* to $\overline{BBPIVER'}$, which proves that both *BBPIVER* and *BBPIVER*$'$ are coNP-hard. In fact, the reduction is the same as in Theorem 9, but here, together with the protocol (as described in the proof of Theorem 9) and the always false specifications, we also return the input assignment $x = (1, n - 2)$ and do not return the integer $k = n - 1$. By looking carefully at the reduction of Theorem 9 it will not be difficult to see that if G has the desired hamiltonian path, then the protocol returned has an erroneous computation when beginning from input x, and if G does not have the desired hamiltonian path, then, for any input (x inclusive), the protocol is correct w.r.t. its specifications. □

5.7.2.6 Alternative Proof of Theorem 9

We have now arrived to an alternative proof that $\overline{BBPVER}$ is NP-hard. The reduction is from $\overline{BBPIVER'}$ to $\overline{BBPVER}$. Given an instance $< \mathcal{A}, \phi, x = (x_0, x_1) >$ of the former we do as follows (keep in mind that we return an instance of the latter of the form $< \mathcal{A}', \phi', k >$). We set $k = x_0 + x_1$, $\phi' = \phi \vee \neg((N_0 = x_0) \wedge (N_1 = x_1))$, and $\mathcal{A}'$ is the union (w.r.t. the output functions) composition of $\mathcal{A}$ and $\mathcal{A}''$ (as in Theorem 10), where $\mathcal{A}''$ is a population protocol for the basic model that stably computes the predicate $\neg((N_0 = x_0) \wedge (N_1 = x_1))$. It is now easy to see (similarly to Theorem 10) that the reduction is correct and can be performed in polynomial time.

5.7.3 An Efficiently Solvable Special Case

We are now seeking for efficiently solvable special cases of the general *GBPVER* problem. A population protocol $\mathcal{A}$ is called *binary* if its input alphabet, its output alphabet, and its set of states are all equal to $\{0, 1\}$. We consider now one of the most trivial cases, which is the *ALLBVER* problem: We are given a binary population protocol $\mathcal{A}$ for the basic model and a formula ϕ representing its specifications. We want again to determine whether $\mathcal{A}$ is always correct w.r.t. ϕ.

The first question that arises is what can ϕ be in this case. So we have to find out first what is stably computable in this simplified model. If the output function of $\mathcal{A}$ is defined as $O(0) = O(1) = y$, where $y \in \{0, 1\}$, then any configuration of $\mathcal{A}$ on any communication graph gives the all-y output. For example, if $y = 0$ then all configurations correspond to the all-zero output, and if $y = 1$ then all configurations correspond to the all-one output. So we have just shown that the trivial predicates (those that are always true or always false) are stably computable. To seek for nontrivial stably computable predicates we have to agree that $O(0) = 0$ and $O(1) = 1$ (this is w.l.o.g. since the case $O(0) = 1$ and $O(1) = 0$ is symmetric). Moreover, we agree that $(0, 0) \rightarrow (0, 0)$ and $(1, 1) \rightarrow (1, 1)$ in the transition function δ. To see this, notice that a nontrivial predicate is true for some inputs and false for others. This means that a protocol for the predicate must be able to stabilize to both the all-zero and the all-one output, and this cannot hold in the absence of the above rules.

Now what about the input function I? Clearly, if $I(0) = I(1) = q$ then the initial configuration is always the all-q configuration (all agents are in state q). For example, if $q = 0$ then the initial configuration is for any input the all-zero configuration. But because of the rules $(0, 0) \rightarrow (0, 0)$ and $(1, 1) \rightarrow (1, 1)$ the population can never escape from its initial configuration, and this case again corresponds to trivial predicates. So, we again agree w.l.o.g. that $I(0) = 0$ and $I(1) = 1$.

It suffices to check the predicates that are stably computable by different combinations of right hand sides for the left hand sides $(0, 1)$ and $(1, 0)$ in δ. There are only 4^2 such combinations so our job is easy. We have the following cases:

- Both $\delta(0, 1)$ and $\delta(1, 0)$ do not belong to $\{(0, 0), (1, 1)\}$. Assume that an input assignment contains one 1 and all other agents get 0. In this case no interaction can increase or decrease the number of 1s so the population forever remains to an unstable configuration (not all agents agree on their output value). So there is no additional stably computable predicate from this case.
- Only one of $\delta(0, 1)$ and $\delta(1, 0)$ belongs to $\{(0, 0), (1, 1)\}$. If one of them is $(0, 0)$ then (since the other offers nothing) if there is at least one 0 in the initial configuration (which is identical to the input assignment, because $I(0) = 0$ and $I(1) = 1$) the protocol rejects, whereas if all inputs are 1 the protocol accepts. We can call this the AND protocol corresponding to the stably computable predicate $\neg(N_0 \geq 1)$. Similarly, if one of them is $(0, 0)$ then we have one form of the OR protocol (see e.g., [16]) and the stably computable predicate corresponding to it is $(N_1 \geq 1)$.
- Both $\delta(0, 1)$ and $\delta(1, 0)$ belong to $\{(0, 0), (1, 1)\}$ and $\delta(0, 1) \neq \delta(1, 0)$. In this case the protocol is unstable. Imagine an initial configuration with exactly one 1 (all other agents get 0) and say that $\delta(0, 1) = (0, 0)$ and $\delta(1, 0) = (1, 1)$. If the unique 1 interacts as the initiator with all other agents in state 0 (one after the other), the protocol in each step replaces a 0 with a 1 and in N_0 steps the population stabilizes to the all-one output. One the other hand if 1 had interacted as the responder with all other agents in the same way as before, then the population would have stabilized to that all-zero output and the protocol is obviously unstable.
- Both $\delta(0, 1)$ and $\delta(1, 0)$ belong to $\{(0, 0), (1, 1)\}$ and $\delta(0, 1) = \delta(1, 0)$. It is easy to see that again we get alternative versions of the OR protocol and the AND

protocol, thus the stably computable predicates resulting from this (last) case are again $\neg(N_0 \geq 1)$ and $(N_1 \geq 1)$.

So we have arrived to a complete characterization of the class of stably computable predicates for the binary basic population protocol model. They are the predicates: always-true, always-false, $\neg(N_0 \geq 1)$, and $(N_1 \geq 1)$.

So we require the specifications ϕ, in the *ALLBVER* problem, to be a stably computable predicate of the binary basic population protocol model, i.e. one of always-true, always-false, $\neg(N_0 \geq 1)$, and $(N_1 \geq 1)$. Obviously, if a binary protocol $\mathcal{A}$ errs on G_2 (the complete graph of 2 nodes) w.r.t. ϕ then it errs in general. But we can also prove that if it errs on some G_k where $k > 2$ then it must err also on G_2 (an easy way to get convinced is to check the statement for all possible classes of protocols as outlined above). This indicates an obvious constant-time algorithm: The transition graph consists of three configurations. For every possible initial configuration find all the final strongly connected components that are reachable from it. If all configurations of those components give the correct output w.r.t. ϕ and this holds for all possible initial configurations, then $< \mathcal{A}, \phi >$ belongs to *ALLBVER*; otherwise $< \mathcal{A}, \phi > \notin$ *ALLBVER*.

5.7.4 Algorithmic Solutions for BPVER

Since Theorem 8 established the coNP-hardness of *BPVER* (Problem 3), our only hope is to devise always-correct algorithms whose worst-case running-time will not be bounded by a polynomial in the size of the input, or algorithms that are not always correct, but are, in fact, correct most of the time (the notion of "approximation" seems to be irrelevant here). Before proceeding, we strongly suggest that the reader carefully revises the definitions from Sect. 5.7.1.

Our algorithms are search algorithms on the transition graph G_r. The general idea is that a protocol $\mathcal{A}$ does not conform to its specifications ϕ on k agents if one of the following criteria is satisfied:

1. $\phi(c) = -1$ for some $c \in C_I$.
2. $\exists c, c' \in C_I$ such that $c \overset{*}{\rightarrow} c'$ and $\phi(c) \neq \phi(c')$.
3. $\exists c \in C_I$ and $c' \in C_F$ such that $c \overset{*}{\rightarrow} c'$ and $O(c') = -1$.
4. $\exists c \in C_I$ and $c' \in C_F$ such that $c \overset{*}{\rightarrow} c'$ and $\phi(c) \neq O(c')$.
5. $\exists B' \in F_S$ such that $O(B') = -1$.
6. $\exists B \in I_S$ and $B' \in F_S$ such that $B \overset{*}{\rightarrow} B'$ and $\phi(B) \neq O(B')$ (possibly $B = B'$).

Note that any algorithm that correctly checks some of the above criteria is a possibly *noncomplete verifier*. Such a verifier guarantees that it can discover an error of a specific kind; thus, we can always trust its "reject" answer (the protocol has some error of this kind). On the other hand, an "accept" answer is a weaker guarantee, in the sense that it only informs that the protocol does not have some error of this specific kind. Of course, it is possible that the protocol has other errors, violating criteria that are undetectable by this verifier. However, this is a first sign of *BPVER*'s *parallelizability*.

Theorem 12 *Any algorithm that checks criteria* 1, 5, *and* 6 *decides BPVER.*

Exercise 15 Prove Theorem 12.

5.7.4.1 Constructing the Transition Graph

Let $FindC_I(\mathcal{A}, k)$ be a function that, given a PP $\mathcal{A}$ and an integer $k \geq 2$, returns the set C_I of all initial configurations. This is not so hard to be implemented. $FindC_I$ simply iterates over the set of all input assignments $\mathcal{X}$ and for each $x \in \mathcal{X}$ computes $I(x)$ and puts it in C_I. Alternatively, computing C_I is equivalent to finding all distributions of indistinguishable objects (agents) into distinguishable slots (initial states), and, thus, Fenichel's algorithm [24] can be used for this purpose.

Algorithm 1 $ConG_r$

Input: PP $\mathcal{A}$ and integer $k \geq 2$.
Output: The transition graph G_r.

```
1:  $C_I \leftarrow FindC_I(\mathcal{A}, k)$
2:  $C_r \leftarrow \emptyset$
3:  $E_r \leftarrow \emptyset$
4:  while $C_I \neq \emptyset$ do
5:      Pick a $c \in C_I$, $C_I \leftarrow C_I - \{c\}$
6:      $C_r \leftarrow C_r \cup \{c\}$
7:      for all $r \in \Delta$ for which $c_i \geq r_{1,2}(i)$ and all $i \in [|Q|]$ for which $q_i \in \{r_1, r_2\}$ do
8:          Compute the unique configuration $c'$ for which $c \xrightarrow{r} c'$.
9:          if $c' \notin C_r$ then
10:             $C_I \leftarrow C_I \cup \{c'\}$
11:         end if
12:         $E_r \leftarrow E_r \cup (c, c')$
13:     end for
14: end while
15: return $(C_r, E_r)$
```

The transition graph G_r can be constructed by the procedure $ConG_r$ (Algorithm 1), that takes as input a population protocol $\mathcal{A}$ and the population size k, and returns the transition graph G_r. The order in which configurations are put in and picked out of C_I determines whether BFS or DFS is used.

5.7.4.2 Noncomplete Verifiers

Now, that we know how to construct the transition graph, we can begin constructing some noncomplete verifiers (which are the easiest). In particular, we present two verifiers, *SinkBFS* and *SinkDFS*, that check all criteria but the last two. Both are presented via procedure *SinkVER* (Algorithm 2) and the order in which configurations of G_r are visited determines again whether BFS or DFS is used.

5.7.4.3 *SolveBPVER*: A Complete Verifier

We now construct the procedure *SolveBPVER* (Algorithm 3) that checks criteria 1, 5, and 6 (and also 2 for some speedup) presented in the beginning of Sect. 5.7.4, and, thus, according to Theorem 12, it correctly solves *BPVER* (i.e., it is a complete verifier for basic population protocols, when the population size is provided as part of the input). In particular, *SolveBPVER* takes as input a PP $\mathcal{A}$, its specifications ϕ and an integer $k \geq 2$, as outlined in the *BPVER* problem description, and returns "accept" if the protocol is correct w.r.t. its specifications on G^k and "reject" otherwise.

Algorithm 2 *SinkVER*

Require: A population protocol $\mathcal{A}$, a Presburger arithmetic formula ϕ, and an integer $k \geq 2$.
Output: ACCEPT if $\mathcal{A}$ is correct w.r.t. its specifications and the criteria 1,2,3, and 4 on G^k and REJECT otherwise.

```
1: $C_I \leftarrow FindC_I(\mathcal{A}, k)$
2: if there exists $c \in C_I$ such that $\phi(c) = -1$ then
3:     return REJECT // Criterion 1 satisfied
4: end if
5: $G_r \leftarrow ConG_r(\mathcal{A}, k)$
6: for all $c \in C_I$ do
7:     Collect all $c'$ reachable from $c$ in $G_r$ by BFS or DFS.
8:     while searching do
9:         if one $c'$ is found such that $c' \in C_F$ and $(O(c') = -1$ or $\phi(c) \neq O(c'))$ then
10:            return REJECT // Criterion 3 or 4 satisfied
11:        end if
12:        if one $c'$ is found such that $c' \in C_I$ and $\phi(c) \neq \phi(c')$ then
13:            return REJECT // Criterion 2 satisfied
14:        end if
15:    end while
16: end for
17: return ACCEPT // Tests for criteria 1,2,3, and 4 passed
```

The algorithmic idea is based on the use of Tarjan's [36] or Cheriyan–Mehlhorn's and Gabow's [20, 25] (or any other) algorithm for finding the strongly connected components of G_r. In this manner, we obtain a collection S, where each $B \in S$ is a strongly connected component of G_r, that is, $B \subseteq C_r$. Given S we can easily compress G_r w.r.t. its strongly connected components as follows. The compression of G_r is a dag $D = (S, A)$, where $(B, B') \in A$ if and only if there exist $c \in B$ and $c' \in B'$ such that $c \to c'$ (that is, iff $B \to B'$). In words, the node set of D consists of the strongly connected components of G_r and there is a directed edge between two components of D if a configuration of the second component is reachable in one step from a configuration in the first one.

Algorithm 3 *SolveBPVER*

Require: A population protocol $\mathcal{A}$, a Presburger arithmetic formula ϕ, and an integer $k \geq 2$.
Output: ACCEPT if the protocol is correct w.r.t. its specifications on G^k and REJECT otherwise.

```
1:  $C_I \leftarrow FindC_I(\mathcal{A}, k)$
2:  if there exists $c \in C_I$ such that $\phi(c) = -1$ then
3:      return REJECT
4:  end if
5:  $G_r \leftarrow ConG_r(\mathcal{A}, k)$
6:  Run one of Tarjan's or Gabow's algorithms to compute the collection $S$ of all strongly con-
    nected components of the transition graph $G_r$.
7:  Compute the dag $D = (S, A)$, where $(B, B') \in A$ (where $B \neq B'$) if and only if $B \to B'$.
8:  Compute the collection $I_S \subseteq S$ of all connected components $B \in S$ that contain some initial
    configuration $c \in C_I$ and the collection $F_S \subseteq S$ of all connected components $B \in S$ that have
    no outgoing edges in $A$, that is, all final strongly connected components of $G_r$.
9:  for all $B \in F_S$ do
10:     if $O(B) = -1$ then
11:         return REJECT
12:     end if
13:     // Otherwise, all configurations $c \in B$ output the same value $O(B) \in \{0, 1\}$.
14: end for
15: for all $B \in I_S$ do
16:     if there exist initial configurations $c, c' \in B$ such that $\phi(c) \neq \phi(c')$ then
17:         return REJECT
18:     else
19:         // all initial configurations $c \in B$ expect the same output $\phi(B) \in \{0, 1\}$.
20:         Run BFS or DFS from $B$ in $D$ and collect all $B' \in F_S$ s.t. $B \overset{*}{\to} B'$ (possibly
            including $B$ itself).
21:         if there exists some reachable $B' \in F_S$ for which $O(B') \neq \phi(B)$ then
22:             return REJECT
23:         end if
24:     end if
25: end for
26: return ACCEPT
```

5.8 Open Problems

The following are some open problems for the interested reader:

- What is the computational power of the variation of the population protocol model in which the agents interact in groups of $k > 2$ agents and not in pairs?
- Recent (unpublished for the time being) research shows that *SPACE(n)* (that is, *LINSPACE*) is a lower bound for the class of symmetric predicates that are stably computable by the basic MPP model, which may be possibly improved to *NSPACE(n)* by exploiting the nondeterminism inherent in the interaction pattern. On the other hand, as mentioned in Sect. 5.3, the best known upper bound is *NSPACE(m)*, and, since we are dealing with complete communication graphs, it holds that $m = \mathcal{O}(n^2)$, which, clearly, leaves a huge gap between the two

bounds. It is possible that $NSPACE(n \log n)$ is a better upper bound. But we do not expect this to be easy, because it would require to prove that we can *encode* the $\mathcal{O}(n^2)$ sized configurations of MPP by new configurations of $\mathcal{O}(n \log n)$ size whose transition graph is, in some sense, *isomorphic* to the old one (e.g., the new configurations reach the same stable outputs). Thus, an exact characterization of this class is still open.

- Is the mediated population protocol model fault tolerant? What are the necessary preconditions to obtain satisfactory fault tolerance?
- Is there an exact characterization of the class of decidable graph languages by MPP in the weakly connected case?
- Is the PALOMA model fault tolerant? What are the necessary preconditions to obtain satisfactory fault tolerance?
- Are there hierarchy theorems concerning all possible models of passively mobile communicating devices? For example, what is the relationship between MPP's class of computable predicates and *PLM*?
- [12] revealed the need for population protocols to have adaptation capabilities in order to keep working correctly and/or fast when natural modifications of the mobility pattern occur. However, we do not know yet how to achieve *adaptivity*.
- Are there more efficient, possibly logic-based, verification solutions for population protocols? Verifying methods for MPPs, Community Protocols, and PALOMA protocols are still totally unknown, although the ideas of Sect. 5.7 may also be applicable to these models.

References

1. D. Angluin, J. Aspnes, M. Chan, M. J. Fischer, H. Jiang, and R. Peralta. Stably computable properties of network graphs. In *Proceedings Distributed Computing in Sensor Systems: 1st IEEE International Conference*, pages 63–74, Marina del Ray, California, USA, 2005.
2. D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006. Also in *23rd Annual ACM Symposium on Principles of Distributed Computing* (*PODC*), pages 290–299, ACM, New York, NY, USA, 2004.
3. D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Urn automata. *Technical Report YALEU/DCS/TR-1280*, Yale University Department of Computer Science, Nov. 2003.
4. D. Angluin, J. Aspnes, and D. Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, September 2008.
5. D. Angluin, J. Aspnes, and D. Eisenstat. Stably computable predicates are semilinear. In *Proc. 25th Annual ACM Symposium on Principles of Distributed Computing*, pages 292–299, Denver, Colorado, USA, 2006.
6. D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4): 279–304, November 2007.
7. J. Aspnes and E. Ruppert. An introduction to population protocols. *Bulletin of the European Association for Theoretical Computer Science*, 93:98–117, October, 2007. Columns: *Distributed Computing*, Editor: M. Mavronicolas.
8. R. Bakhshi, F. Bonnet, W. Fokkink, and B. Haverkort. Formal analysis techniques for gossiping protocols. In *ACM SIGOPS Operating Systems Review*, 41(5):28–36, Special Issue on Gossip-Based Networking, October 2007.

9. J. Beauquier, J. Clement, S. Messika, L. Rosaz, and B. Rozoy. Self-stabilizing counting in mobile sensor networks. *Technical Report 1470*, LRI, Université Paris-Sud 11, 2007.
10. G. Behrmann, A. David, and K. G. Larsen. A tutorial on Uppaal. In *Formal Methods for the Design of Real-Time Systems: Proceedings of the 4th International School on Formal Methods for the Design of Comput., Commun. and Software Syst. (SFM-RT 2004)*, number 3185 in LNCS, pages 200–236, Springer, 2004.
11. O. Bournez, P. Chassaing, J. Cohen, L. Gerin, and X. Koegler. On the convergence of population protocols when population goes to infinity. In *Applied Mathematics and Computation*, 215(4):1340–1350, 2009.
12. I. Chatzigiannakis, S. Dolev, S. P. Fekete, O. Michail, and P. G. Spirakis. Not all fair probabilistic schedulers are equivalent. In *13th International Conference On Principles Of DIstributed Systems (OPODIS)*, pages 33–47, Nimes, France, December 15–18, 2009.
13. I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P. G. Spirakis. Algorithmic verification of population protocols. FRONTS Technical Report FRONTS-TR-2010-12, http://fronts.cti.gr/aigaion/?TR=148, Jan. 2010.
14. I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P. G. Spirakis. Passively mobile communicating logarithmic space machines. FRONTS Technical Report FRONTS-TR-2010-16, http://fronts.cti.gr/aigaion/?TR=154, Feb. 2010.
15. I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Decidable graph languages by mediated population protocols. In *23rd International Symposium on Distributed Computing (DISC)*, Elche, Spain, Sept. 2009. (Also FRONTS Technical Report FRONTS-TR-2009-16, http://fronts.cti.gr/aigaion/?TR=80)
16. I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Experimental verification and performance study of extremely large sized population protocols. FRONTS Technical Report FRONTS-TR-2009-3, http://fronts.cti.gr/aigaion/?TR=61, Jan. 2009.
17. I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Mediated population protocols. In *36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 363–374, Rhodes, Greece, 2009.
18. I. Chatzigiannakis, O. Michail, and P. G. Spirakis. Recent advances in population protocols. In *34th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, August 24–28, 2009, Novy Smokovec, High Tatras, Slovakia.
19. I. Chatzigiannakis and P. G. Spirakis. The dynamics of probabilistic population protocols. In *Distributed Computing*, *22nd International Symposium*, *DISC*, vol. 5218, *Lecture Notes in Computer Science*, pages 498–499, 2008.
20. J. Cheriyan and K. Mehlhorn. Algorithms for dense graphs and networks on the random access computer. *Algorithmica*, 15: 521–549, 1996.
21. E. M. Clarke, O. Grumberg, and D. A. Peled. Model checking. MIT Press, New York, NY, 2000.
22. C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. When birds die: Making population protocols fault-tolerant. In *Proc. 2nd IEEE International Conference on Distributed Computing in Sensor Systems*, pages 51–66, 2006.
23. Z. Diamadi and M. J. Fischer. A simple game for the study of trust in distributed systems. *Wuhan University Journal of Natural Sciences*, 6(1-2):72–82, Mar. 2001. Also appears as Yale Technical Report TR-1207, Jan. 2001.
24. R. Fenichel. Distribution of Indistinguishable Objects into Distinguishable Slots. *Communications of the ACM*, 11(6): 430, June 1968.
25. H. N. Gabow. Path-based depth-first search for strong and biconnected components. *Information Processing Letters*, 74:107–114, 2000.
26. D. T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A*, 188:404–425, 1992.
27. D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

28. S. Ginsburg and E. H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific Journal of Mathematics*, 16:285–296, 1966.
29. R. Guerraoui and E. Ruppert. Names trump malice: Tiny mobile agents can tolerate byzantine failures. In *36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 484–495, Rhodes, Greece, 2009.
30. A. Hinton, M. Z. Kwiatkowska, G. Norman, and D. Parker. Prism: A tool for automatic verification of probabilistic systems. In *Proceedings of 2nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '06)*, vol. 3920, LNCS, pages 441–444. Springer, 2006.
31. G. Holzmann. *The Spin Model Checker, Primer and Reference Manual*. Addison-Wesley, New York, NY, 2003.
32. M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, Cambridge, UK, 2004.
33. T. G. Kurtz. Approximation of population processes. Number 36 in *CBMS-NSF Regional Conference Series in Applied Mathematics*, *Society for Industrial and Applied Mathematics*, Philadelphia, 1981.
34. P. C. Olveczky and S. Thorvaldsen. Formal modeling and analysis of wireless sensor network algorithms in Real-Time Maude. *Parallel and Distributed Processing Symposium, International*, p. 157, *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, Rhodes, Greece, 2006.
35. C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, New York, NY, 1994.
36. R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

Chapter 6
Theoretical Aspects of Graph Models for MANETs

Josep Díaz, Dieter Mitsche, and Paolo Santi

Abstract We survey the main theoretical aspects of models for mobile ad hoc networks (MANETs). We present theoretical characterizations of mobile network structural properties, different dynamic graph models of MANETs, and finally we give detailed summaries of a few selected articles. In particular, we focus on articles dealing with connectivity of mobile networks and on articles which show that mobility can be used to propagate information between nodes of the network while at the same time maintaining small transmission distances and thus saving energy.

6.1 Introduction

In 1961 Edward Gilbert [21] defined *random plane networks* as a model to study the communication in networks of short-range stations spread over a large area. In his model, vertices represent the stations, and edges represent a two-way communication channel between stations. All stations have the same range power, so there is a direct communication between two stations iff the corresponding vertices are connected by an edge. Gilbert distributed the vertices in an infinite plane, by using a Poisson point process in the plane and then connecting two vertices if they are separated by at most a distance r. He went to study the asymptotic value of the probability that a vertex belongs to a connected component with all the other vertices.

Nowadays, Gilbert's model is better known as *random geometric graphs* (RGG). A random geometric graph can be equivalently defined by distributing n points uniformly on a given surface; thus, a RGG is a graph resulting from placing a set of n vertices independently and uniformly at random on the unit square $[0, 1]^2$, and by connecting two vertices if and only if their *distance* is at most the given radius r, the distance depending on the type of metric being used. For convenience, when using

J. Díaz (✉)
Departament de Llenguatges, Sistemes i Informàtics, Universitat Politècnica de Catalunya Grup de Recerca ALBCOM, Barcelona, Spain
e-mail: diaz@lsi.upc.edu

a Poisson point process to distribute the vertices, sometimes it is better to scatter the vertices on $[\sqrt{n}, \sqrt{n}]^2$, where n is the expected number of points distributed by the Poisson process. It is well known that the results in this model are just rescaled versions of the results on $[0, 1]^2$. Some authors consider the torus $[0, 1)^2$ to avoid the effects of boundaries, which we will mention in Sect. 6.3 in more detail. For many properties the boundary effects change the results, see, for example, [62]. We refer to an instance of a RGG with n vertices and radius r as $\mathcal{G}(n, r)$.

The deterministic counterparts of random geometric graphs are called *unit disk graphs* (UDG). A graph G is a *unit disk graph* for a radius distance r, if its vertices can be put in one-to-one correspondence with the centers of circles of radius r in the plane, in such a way that two vertices in G are connected by an edge if and only if their corresponding circles intersect [12, 24]. The *recognition problem* is to decide whether a given graph G is a UDG. The problem is known to be NP-hard [8]. Since the vertices of a UDG are points in the real plane, the problem is not known to be in NP.

Random geometric graphs and unit disk graphs have received quite a bit of attention in the last years both as a particular mathematical structure different from other types of known graphs [47, 59] and also because of their applications as models for wireless networks, in particular as simplified topological models for *wireless sensor networks* (see, for example, [55], where UDG was also denoted as *point graph model* and [1, 4, 17, 56, 57, 67]). Further applications of unit discs and random geometric graphs as models for more general *ad hoc networks* are discussed in the references [14, 29, 31, 42, 66] and in Chapter 1 of [59].

Wireless networks consist of a set of *simple* nodes, each one with a wireless transceiver to communicate with their near neighbors, where *near* is understood as the closest in terms of Euclidean distance, and the ability of communication depends on the transmitting power of the transceivers. The goal of a network is to spread information through the network, which is done in a multi-hop fashion. In many ad hoc networks, like sensor networks, due to the simplicity of the nodes, energy consumption is an issue. Therefore, one of the most important questions when modeling a network is to minimize power consumption. That is, the transmission range should be made as small as possible but at the same time large enough to make sure that a packet of information transmitted from a node will arrive to the other nodes in the network. As we mention in the next section, when modeling wireless networks by graph topology, one of the main problems is the trade-off between range of transmission and network connectivity. In Sect. 6.5.3 we will give examples where mobility boosts message distribution in a network while at the same time maintaining a small range of transmission.

The choice of whether to use a deterministic model, such as UDG, or a randomized model, such as RGG, depends on the application. For example, when using sensors networks, it is usual that the sensors are scattered from some type of vehicle and hence in this case the random model is the appropriate one. For other kinds of wireless networks, the randomized model also could be interesting to obtain the *average behavior* of the network. In the next section, we also briefly mention the case where the transmitting power of each node is different, introducing the *range* *assignment* problem: the problem of assigning different transmission powers to each node in such a way that the power used is minimized, while maintaining the network connected.

The aim of the present work is to survey the recent theoretical results for *mobile ad hoc networks* (MANETs), with an emphasis of topological models. It is organized as follows: in Sect. 6.2 we review a few known results about static RGG, mainly those related to connectivity; in Sect. 6.3 we discuss issues that play a role in dynamic models and present different random mobility models; in Sect. 6.4 we survey theoretical results concerning a very popular mobility model (the random waypoint model), showing in particular how mathematical tools have been used to identify problems in wireless mobile network simulation and to solve them; in Sect. 6.5, we present a few selected recent papers on dynamic MANETs, focusing on papers which present a formal analysis of mobility model properties and use the analysis to characterize fundamental network properties such as connectivity and information propagation speed. Throughout this chapter, "a.a.s." denotes *asymptotically almost surely*, that is, with probability tending to 1, as n goes to ∞. For other concepts in probability, the reader is advised to look into any of the basic references, for example, [26, 48].

6.2 Static Properties

In this section, we point out some of the known results about *static* RGG, which will be helpful for the mobility survey. In this line, we skip many of the very interesting recent results on RGG that are of combinatorial nature, such as results about the chromatic number, for example. The main reference on RGG is the book by Mathew Penrose [47]. Moreover, the reader should be aware that since 2002 a lot of work has been done on the topic of static RGG. When considering a RGG as topological model for a wireless network, one of the important issues is to keep the network connected using the minimal amount of energy consumption, i.e., using the smallest transmitting distance. This is called the *critical power* among the networking community [42] and the *connectivity threshold* among the mathematical community [47]. In the book of Penrose, the results are exposed in full generality, for any distance norm and any dimension. To make the basic ideas as clear as possible, in the present survey we stick to the case of dimension 2 and Euclidean distance norm.

Let $\mathcal{G}(n, r)$ be the graph representing a wireless ad hoc network with n nodes, where r denotes the transmitting distance. We assume the ideal case where the area covered by a node is exactly a circle. *Topology control* is a technique that uses the tuning of certain parameters, usually the transmitting range r or the maximum degree of the graph, to change/form the topology of the graph representing the network in order to maintain the connectivity while optimizing the energy (or minimizing the interference). There are very good recent surveys on the topic of topology control, see, for instance, [42, 52]. One of the important problems in topology control is the critical transmitting range for connectivity: *what is the smallest radius,* *denoted by* r_c, *that keeps* G *connected?* If G is a deterministic instance, i.e., a UDG, it is well known that the value of r_c is the length of the longest edge in the *minimum spanning tree* (MST) of G. The case where G is a RGG is more interesting. In this case, Penrose [45] computed the expected length of the longest edge of a MST in a RGG on $[0, 1]^2$, yielding the well-known connectivity threshold $r_c = \sqrt{\frac{\ln n \pm O(1)}{\pi n}}$ a.a.s., where as usual the abbreviation a.a.s. stands for *asymptotically almost surely*, i.e., with probability $1 - o(1)$ as $n \to \infty$. Independently, [22] gave the same bounds for r_c, also for the ℓ_2-norm but considering the unit circle as underlying surface. Both proofs are quite different.

Notice that real wireless networks cannot be too dense, because a transmitting node interferes with all the nodes within its interference range. In [53, 54] the authors have characterized the critical transmission range in the more general model in which the side ℓ of the deployment region is a further parameter, and n and r can be arbitrary functions of ℓ. Note that under this model, the asymptotic behavior of node density (number of nodes per unit area) depends on how n changes with ℓ. In particular, n can be chosen in such a way that node density asymptotically converges to 0, or to an arbitrary constant greater than 0, or diverge. Under this respect, Santi et al.'s model is more general than the standard RGG model, in which node density grows to infinity with n. The main finding of [53, 54] is a proof that, as $\ell \to \infty$, if $r \sim \ell\sqrt{c\frac{\lg \ell}{n}}$ for some constant $c > 0$, then the graph is connected a.a.s.

Going back to the classical model of RGG on $[0, 1]^2$, we now try to convey the flavor and intuition behind the value r_c for which a RGG becomes connected a.a.s. Given a set V of n nodes and a positive real $r = r(n)$, each node is placed at some random position in $[0, 1]^2$ selected uniformly at random. We define $\mathcal{G}(n, r)$ as the random graph having V as the vertex set, and with an edge connecting each pair of vertices u and v at distance $d(u, v) \leq r$, where $d(\cdot, \cdot)$ denotes the Euclidean distance. We assume that $r = o(1)$, else $\mathcal{G}(n, r)$ is trivially connected a.a.s. Let X be the random variable counting the number of isolated vertices in $\mathcal{G}(n, r)$. Then, by multiplying the probability that one vertex is isolated by the number of vertices we obtain

$$\mathbf{E}(X) = n(1 - \pi r^2)^{n-1} = ne^{-\pi r^2 n - O(r^4 n)}$$

Define $\mu = ne^{-\pi r^2 n}$. Observe that this parameter μ is closely related to $\mathbf{E}(X)$. In fact, $\mu = o(1)$ iff $\mathbf{E}(X) = o(1)$, and if $\mu = \Omega(1)$ then $\mathbf{E}(X) \sim \mu$.

Moreover, the asymptotic behavior of μ characterizes the connectivity of $\mathcal{G}(n, r)$. In fact, if $\mu \to 0$, then a.a.s. $\mathcal{G}(n, r)$ is connected; if $\mu = \Theta(1)$, then a.a.s. $\mathcal{G}(n, r)$ consists of one giant component of size $> n/2$ and a number of isolated vertices which follows a Poisson distribution with parameter μ; and if $\mu \to \infty$, then a.a.s. $\mathcal{G}(n, r)$ is disconnected. Therefore, from the definition of μ we have that $\mu = \Theta(1)$ iff $r_c = \sqrt{\frac{\ln n \pm O(1)}{\pi n}}$ (see [47]).

Extensions to k-connectivity appear in [46], where the author proves that when the minimum degree of a RGG is k the graph becomes k-connected. Notice that

k-connectivity is important in networking as a measure of fault tolerance of the network. Chapter 13 of [47] presents an extensive treatment of connectivity for RGG, taking into account different norms, higher dimensions, and different underlying probability distributions.

Recall that a graph property is *monotone* if it is preserved when edges are added to the graph. A graph property is said to have a *sharp threshold* if the window between having and not having the property can be made arbitrarily small. In [25] the authors prove that every monotone property on a RGG has a *sharp threshold*. As connectivity is a monotone property, we conclude that the property of connectivity in $\mathcal{G}(n, r)$ exhibits a sharp threshold at $r_{\mathrm{c}} = \sqrt{\frac{\ln n}{\pi n}}$.

As mentioned before, for a radius r slightly below the connectivity threshold r_{c}, $\mathcal{G}(n, r)$ consists a.a.s. of a giant component and some isolated vertices. It is also known that in this situation the probability of having a component of size i at r_{c} is $O(1/\log^i n)$, and, if there exists one, it forms a clique [16]. A straightforward computation yields that when we consider the *connectivity regime* with $r = r_{\mathrm{c}}$, the expected degree of a vertex is asymptotically $\Theta(\log n)$ (plug r_{c} in the expected number of neighbors of a vertex, which is $\pi r_c^2(n-1)$). For values of $r > r_{\mathrm{c}}$, $\mathcal{G}(n, r)$ is said to be in the *superconnectivity regime* and the graph is *dense*,[1] while for values of $r < r_c$, $\mathcal{G}(n, r)$ is said to be in the *subconnectivity regime* and the graph is *sparse*. As we mention in Sect. 6.5.3, in the subconnectivity regime mobility can help to spread information.

The behavior of RGG for values of r in the subconnectivity regime has been quite thoroughly studied, see Chapter 10 in [47]. It is known that there exists a value $r_t = \frac{c}{\sqrt{n}}$ where a giant component of size $\Theta(n)$ appears in $\mathcal{G}(n, r)$ a.a.s., with c being a constant that experimentally is conjectured to have a value around 2.35 (recall that we focus on the ℓ_2-norm in two dimensions). In the regime where $r < r_{\mathrm{t}}$, each vertex has expected degree $O(1)$. The r_{t} is denoted as the *thermodynamical threshold*.

The *cover time* C of $\mathcal{G}(n, r)$ is the expected time taken by a simple random walk of $\mathcal{G}(n, r)$ to visit all the nodes in the graph. In [2] the authors prove that a.a.s. $C = \Theta(n \log n)$ if $r \geq \sqrt{\frac{c \log n}{n}}$, with $c > 8$. If $r \leq \sqrt{\frac{\log n}{\pi n}}$, the cover time is ∞ with positive probability, bounded away from zero.

When dealing specifically with *wireless sensor networks*, an important issue is to assure that sensors properly cover the entire region being monitored, which is known as the *coverage* of the network. Similar to connectivity, coverage can be modeled using the RGG model, where each vertex represents a sensor and r is the *sensing range* of the sensors. Given an integer k, a point is said to be *k-covered* if it falls into the sensing range of at least k sensors. If all the points of a region are k-covered, then the region is k-covered. If $\mathcal{C}_{n,r}^k$ denotes the event that every point of $[0, 1]^2$ is $(k+1)$-covered by a network with n sensors of range r, the *k-covering problem* consists in giving asymptotic bounds to $\mathbf{Pr}\left[\mathcal{C}_{n,r}^k\right]$, as $n \to \infty$. In [30], the case $k = 1$

[1] Note that a usual graph with n vertices is said to be dense if it has $\Theta(n^2)$ edges.

is studied; however, the author uses a toroidal metric to avoid problems with nodes very near the boundary of the region where the nodes are scattered. Several authors have been working on this problem [40, 44, 61, 68]. In [62] the authors give bounds on $\mathbf{Pr}\left[\mathcal{C}_{n,r}^{k}\right]$ for the unit square, taking into consideration the boundary effect of the unit square, which complicates quite a bit the analytical proof. Sometimes, coverage and connectivity of a wireless sensor network are jointly studied, with the objective of forming a network which not only k-covers the entire monitored region but also is connected. It is easy to see that k-coverage implies k-connectivity of the network whenever $r_{\mathrm{t}} \geq 2r_{\mathrm{s}}$, where r_{t} is the transmission range and r_{s} is the sensing range of nodes [63].

Up to now we have considered that all nodes broadcast at the same transmitting range r, but the efficiency of energy management in a network could be achieved by tuning every node to a different transmitting range. The *range assignment* problem is the following: given a graph with n nodes, each one knowing their position, the goal is to assign a transmitting range r_i to each node i in such a way that the network is connected with minimum energy cost, where the energy e_i used by node i is proportional to r_i^2, i.e., the goal is to minimize $\sum_i e_i$. The problem was first studied in [39]. Since then, several authors have proposed and studied different variants of the basic model, see Section 5.5.2 in [52].

Another important issue is the design of efficient protocols for disseminating and broadcasting information in wireless ad hoc networks. We refer the reader to one of the multiple surveys treating the topic [34, 42, 49, 50, 52, 66].

6.3 Mobility Models for MANETs

After giving a very concise introduction to the results on static random geometric graphs, let us focus our attention on mobility issues. When talking about mobility in MANETs, we mean mobility of the nodes, i.e., the nodes physically move in a region. There is an alternative version of dynamical wireless networks, where the dynamicity is caused by the addition and removal of edges between nodes, due to the temporal evolution of the transmitting range r_i, for each node i. This kind of mobility has been thoroughly studied by the computational geometry community, see, for example, [3, 20]. The main focus of their research is the design and analysis of sophisticated algorithms and data structures that easily allow deletion or addition of very few edges or nodes at each time. In the case of highly dynamic MANETs, due to the large number of changes in each step, the direct evaluation of the performance of the network is very time consuming (see, for example, Section 2 in [5]). One way to get an idea of the performance is to use simplified models of the network. Moreover, due to the fact that real MANETs are mostly deployed in environments where it is difficult to control the quality of transmission, simulation could furnish better scenarios to control the experiments. In particular, when designing new protocols for communication, sometimes it is better to start simulating on a simplified topology than a direct implementation on the real network. However, some researchers reason that low-scale simulations are not conclusive and that the final validation of the viability and efficiency of the new proposed protocol must be experimented directly on the network (see, for example, [41]).

In the remainder of this survey, we are going to look at the recent preliminary research done on analytical studies of different mobility models proposed. The goals of the simplified models are to extract the topological properties of mobile networks, which might help both in improving simulation accuracy (see Sect. 6.4), and in designing new protocols where mobility is used to reduce energy consumption and/or information propagation speed (see Sect. 6.5). Clearly, this survey does not cover *every* property where mobility helps. For example, for the k-covering problem, in [60] the authors recently proved both analytically and experimentally that, if a fraction of the nodes is mobile with very limited range of mobility, k-coverage can be achieved with less sensors than in the static case.

In the last decade, quite a few models for MANETs have been proposed, see the surveys [5, 6, 10, 69]. Section 2.1.5 of [5] gives a detailed taxonomy of the mobility models used in the literature. According to the degree of mobility, there are three types of mobility:

- The *deterministic model* where nodes move through predetermined paths in a deterministic manner. The model needs to trace the mobility of nodes, which can be cumbersome [58].
- The *hybrid random model* where the model guides the nodes through a predetermined graph, which represents streets, roads, etc. On this graph, however, nodes move randomly. For example, in [36] the authors consider a region with obstacles and force the mobility to take place along the Voronoi tessellation of the obstacles. The *city selection mobility* and the *graph-based mobility* models described in [5] are the examples of hybrid random models.
- The *pure random model* where the nodes move in a random way in the region. Most of the models, described in the literature, belong to this class. The two most representative models in this class are the *random direction model* and the *random waypoint model*.

The most frequently used mobility models are the following two and their variations:

- the *random waypoint model* (RWP) was first described in [37]. In this model, as usual, nodes are initially distributed uniformly at random on the region; then, each node chooses independently and uniformly at random a destination within the region as well as a travel speed. The node then starts traveling toward the destination with the selected speed along a linear trajectory. When it reaches the destination (waypoint), it might optionally pause for a certain time, then chooses another waypoint in the region and continues according to the same pattern. Structural properties of RWP model have been deeply investigated in the literature and are discussed in detail in Sect. 6.4.
- the *random direction model* (RD): the seed of the RD model is the paper [28], in which each node i in the region under consideration selects uniformly at random

a direction $\theta_i \in [0, 2\pi)$ and chooses a speed that is kept constant during a certain amount of time. After a randomly chosen period of time, each node selects a new direction and speed and continues moving. As the process evolves over time, some of the nodes might arrive at the boundary of the region, and a *border rule* has to be defined to determine how nodes behave when they hit the border. An easy way to deal with the boundary effect is to consider the toroidal version $[\ell_1, \ell_2)^2$ instead of the unit square $[\ell_1, \ell_2]^2$. In fact, when modeling applications like sensor networks on large terrain, the toroidal model is a fair approximation to reality. For smaller areas, when the boundary effect is significant, an alternative option is to consider the so-called *bouncing boundary rule*, where the nodes arriving at a boundary bounce back to the region. When a node hits the boundary, this bouncing could be done either by choosing a random new angle θ' or by following the *mirror reflection* rule, i.e. the node returns to the region at an angle $\theta' = \pi - \theta$, where θ is the incidence angle at which the node hits the boundary. There have been several modifications of the basic RD model, some of them specifically designed to deal with the border effect [35] (see below for a definition of border effect). The RD model has been criticized because of the unrealistic behavior caused by uncorrelated changes in direction and speed (see, for example, [32]). In [7], the author proposed a variation of the RD model, with two correlated processes, one to define the speed and another one to define the changes in direction (no correlation between different nodes). The authors denoted this variation the *smooth random mobility model*.

Note that the fact that moves in a bounded region gives rise to the so-called *border effect*, which in general can be understood as a modification of the *probability density function (pdf)* describing mobile node positions with respect to the initial *pdf* (typically, uniform) due to the presence of a border. The border effect arises not only in models (such as RD) in which nodes can hit the border and border rules are used to define node behavior in such situation but also in models (such as RWP) in which nodes can never reach the border of the movement region. Further detailed explanation of the border effect in RWP mobile networks is reported in the next section.

Two further models different to the previous ones are the following:

- The *Brownian motion model*: each of the x- and the y-coordinates describing the current position of each node undergoes a continuous-time stochastic process (these processes are independent for both coordinates and independent for all nodes), which is almost surely continuous and the changes in the positions between any two times t_1, t_2 with $0 \leq t_1 \leq t_2$ follow a normal distribution $N(0, t_2 - t_1)$. Moreover, the changes between $t_1 \leq t_2$ are independent from the changes in $t_3 \leq t_4$, if $t_2 \leq t_3$. Brownian motion can be considered as the *limit* case of the random direction model, where the period of time after which a new angle is chosen tends to 0 (see, for example, [11]).
- An approach orthogonal to the previous ones was undertaken in [19] in order to accomplish group communication tasks between a set of processors. The model is the following: given n processors executing programs, the communication

between the processors is established with the help of an agent who visits the processors. If there are more than one agent and two agents collide at one processor, they merge into one, and if there is no agent, after some time an agent is automatically generated by a processor. The agent performs a random walk on the processors (the next processor could be chosen from some suitably defined neighborhood of the current processor or it could be chosen from the whole set of processors), and whenever it arrives at a processor, the processor stops its current program and replaces it by a new program using the information the agent is carrying. The agent's goal is to broadcast the information in such a way that each processor is visited by the agent at least every M steps, where M depends on n, and that each processor executes a step infinitely often. The authors design agents satisfying these conditions for different group communication tasks and they prove that starting from any arbitrarily chosen node, these agents have an expected cover time of at most $O(n^3)$.

6.4 Structural Properties of Random Waypoint Mobile Networks

In this section, we present theoretical characterizations of structural properties of networks whose nodes move according to a very popular mobility model: the *random waypoint* mobility model (RWP). We show that how these characterizations have been used to considerably improve accuracy of wireless network simulation. Some of these characterizations (e.g., node spatial distribution) have been used also to study fundamental mobile network properties, such as connectivity (see Sect. 6.5).

RWP is by far the most commonly used mobility model used in wireless mobile network simulation. Given its popularity, the structural properties of RWP mobile networks have been deeply investigated in the literature as well as their effects on simulation accuracy.

In the remainder of this section, we focus our attention on two such structural properties, namely *node spatial distribution* and *instantaneous average nodal speed*, and discuss their impact on accuracy of RWP mobile network simulation. We then show how theoretical characterizations of the above properties have been used to define a "perfect" simulation methodology, which completely removes the accuracy issues previously identified.

6.4.1 RWP Node Spatial Distribution

The first structural property of RWP mobile networks that has been formally studied is the asymptotic node spatial distribution, which can be formally defined as follows. Let f_t be the *pdf* describing node position within the movement region at time t of the mobility process. The asymptotic node spatial distribution is a *pdf* formally defined as

$$f_\infty = \lim_{t\to\infty} f_t$$

whenever the limit on the right hand side exists, i.e., that the mobility model has a *stationary* node spatial distribution. In the literature, it has been proven that most mobility models described in the previous section (e.g., RWP, RD, Brownian) indeed have a stationary node spatial distribution.

In the following, we present a formal characterization of f_∞ in the presence of RWP node mobility, which we denote by f_{RWP}. In particular, we will survey results that show that $f_{\mathrm{RWP}} \neq f_U$ (f_U is the uniform *pdf* on the movement region), unless the expected pause time at the waypoints tends to infinity.[2] Thus, we are in presence of the border effect, which can cause considerable inaccuracies in wireless network simulation. In fact, if simulation results are gathered after a relatively short time after network setup, the node spatial distribution of RWP mobile nodes might not have reached the stationary condition, implying that, from a topological point of view, network conditions are different from those reached at stationary state. To make this point clearer, assume that results of a network simulation are averaged over a time interval starting after 100 s since the beginning of simulation and ending after 900 s (these are quite standard simulation intervals in the networking literature). Furthermore, assume that RWP node spatial distribution takes 1000 s to stabilize (this is also a reasonable stabilization time, see [9]). Then the outcome of the simulation experiment might be highly inaccurate, since results are gathered *before* the network has reached its stationary state.

Another pitfall of the border effect is on networking protocol performance optimization: typically, networking protocols (e.g., routing protocols) are optimized under the assumption that nodes are uniformly distributed in a certain region. However, if nodes move according to RWP mobility, this assumption is no longer true at stationary state, implying that protocol performance can indeed be highly suboptimal in presence of mobility.

The first analytical study of node spatial distribution under RWP mobility is reported in [9], for the case of nodes moving in the unit square. In that paper, RWP mobility is described as a stochastic process $\{D_i, T_{\mathrm{p},i}, V_i\}$, where D_i is a random variable denoting the two-dimensional coordinates of trip i destination, $T_{\mathrm{p},i}$ is a random variable denoting the pause time at D_i, and V_i is a random variable denoting the node velocity during trip i. The actual value of D_i will be represented by d_i. First, the authors prove a result concerning ergodicity of the sequence of random variables $\{L_i\}$, where $L_i = ||d_i - d_{i-1}||$, that is, L_i denotes the length of the ith trip. In particular, the authors show that repeatedly sampling from a single random variable in the sequence is statistically equivalent to successively sampling from the sequence $\{L_i\}$. This first result allows reducing the problem of characterizing $f_{\mathrm{RWP},0}$ when the pause time at waypoint is 0 to one of computing the intersection between a random trajectory and an arbitrarily small square of side $\delta > 0$ centered

[2] Note that the fact that the expected pause time at waypoints tends to infinity implies that nodes are asymptotically static, i.e., RWP model under this condition degenerates to a static network.

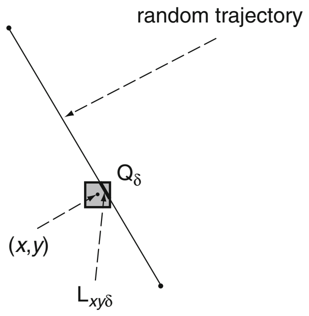


Fig. 6.1 The *pdf* of a RWP mobile node can be characterized by computing the expected length of the segment $L_{xy\delta}$ representing the intersection between a random trajectory and a square Q_δ of side δ centered at (x, y) (*shaded area*)

at a certain coordinate (x, y) (see Fig. 6.1). This stems from the fact that f_{RWP} can be considered as constant within Q_δ as $\delta \to 0$, implying that

$$f_{\text{RWP},0}(x, y) = \lim_{\delta \to 0} \frac{P(x, y, \delta)}{\delta^2} \tag{6.1}$$

where $P(x, y, \delta)$ is the probability that an RWP mobile node is located within a square of side δ centered at (x, y). Thus, $f_{\text{RWP},0}$ can be characterized by evaluating $P(x, y, \delta)$. Since ergodicity of $\{L_i\}$ implies for a successively large sample size that

$$P(x, y, \delta) = \frac{E[L_{xy\delta}]}{E[L]}$$

and $E[L]$ (the expected distance between two random uniform points in a square) is well known from geometric probability, characterizing $f_{\text{RWP},0}$ boils down to computing $E[L_{xy\delta}]$, i.e., the expected length of the intersection between a random trajectory and a square of side δ centered at (x, y). The value of $E[L_{xy\delta}]$ is closely approximated in [9] through computing a set of two-dimensional integrals, yielding the following expression for f_{RWP} in the region $(0 \le x \le 0.5) \cup (0 \le y \le x)$[3]:

$$f_{\text{RWP},0}(x, y) = 6y + \frac{3}{4}\left(1 - 2x + 2x^2\right)\left(\frac{y}{y-1} + \frac{y^2}{x(x-1)}\right) + \\ + \frac{3y}{2}\left[(2x-1)(y+1)\ln\left(\frac{1-x}{x}\right) + (1 - 2x + 2x^2 + y)\ln\left(\frac{1-y}{y}\right)\right]$$

The density function $f_{\text{RWP},0}$ is drawn in Fig. 6.2. As seen from the figure, f_{RWP} is bell shaped with a higher concentration in the center of the movement region,

[3] Values of $f_{\text{RWP},0}$ in the other regions of the unit square are obtained by symmetry.

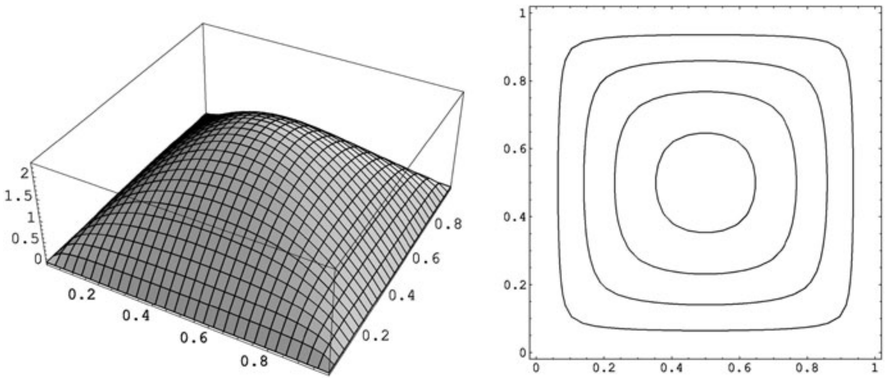


Fig. 6.2 Density function of a RWP mobile network with pause time set to 0: 3D plot (*left*), and contour lines (*right*)

reflecting the fact that a random trajectory is relatively more likely to cross the center than the boundary of the region.

After deriving the *pdf* under the assumption of zero pause time, the authors of [9] consider the more general case of pause times chosen according to an arbitrary probability distribution and show that the resulting node spatial distribution has the following shape:

$$f_{\mathrm{RWP}} = p_p f_U + (1 - p_p) f_{\mathrm{RWP},0}$$

where $p_{\mathrm{p}} = \lim_{t\to\infty} p_{\mathrm{p}}(t)$, and $p_{\mathrm{p}}(t)$ is the probability that an RWP mobile node is pausing at time t. Thus, f_{RWP} is the sum of two components: a uniform component, accounting for the fact that when nodes are resting at a waypoint they are uniformly distributed, and a non-uniform component, reflecting the fact that when nodes are moving they are more likely located near the center of the movement region. The derivation of p_{p} is quite straightforward and yields

$$p_{\mathrm{p}} = \frac{E[T_{\mathrm{p}}]}{E[T_{\mathrm{p}}] + \frac{E[L]}{v}}$$

under the hypothesis that the node velocity is fixed to $v > 0$.

In a more recent paper [33], Hyytiä et al. provide the exact characterization of $f_{\mathrm{RWP},0}$ and generalize the previously described results to arbitrary convex shapes of the movement region and arbitrary waypoint distribution.

6.4.2 RWP Average Nodal Speed

A second property of RWP mobile networks that has been extensively studied is the *average nodal speed*, which is formally defined as follows. Assume n nodes move independently within a region according to the RWP mobility model and denote by $v_i(t)$ the instantaneous speed of the ith node at time t. The asymptotical average nodal speed v_{RWP} is defined as

$$v_{\mathrm{RWP}} = \lim_{t\to\infty} \frac{\sum_{i=1}^{n} v_i(t)}{n}$$

The first paper that formally investigates the average nodal speed in RWP mobile networks is [64], where the authors prove that $v_{\mathrm{RWP}} \neq v_0$ as long as the trip velocity is randomly chosen in a non-degenerate interval and v_0 is the average nodal speed at time 0. Before giving some details of the derivation, we observe that the fact that $v_{\mathrm{RWP}} \neq v_0$ gives rise to the so-called *speed decay phenomenon*, which displays many similarities with the border effect described in the previous section. In fact, similar to border effect, speed decay affects both simulation accuracy and optimization of network protocols, for the very same reasons the border effect did, i.e., (i) stationary conditions for what concerns node velocity are different from initial ones and (ii) they are reached only after a relatively long stabilization period.

The authors of [64] derive v_{RWP} under the following three assumptions:

1. nodes move in an unlimited, arbitrarily large area; given the current node location (x, y), the next waypoint is chosen uniformly at random in a circle of radius $R_{\max}$ centered at (x, y).
2. the pause time is 0.
3. the node velocity is chosen uniformly at random from $[v_{\min}, v_{\max}]$.

While the second and third assumptions are standard, the first assumption, which is done to simplify analysis, apparently perturbs quite a bit the properties of the mobility model. In the paper it is shown that this assumption has no effect on the value of v_{RWP}, which remain the same as in the case of standard, bounded RWP mobility.

Similar to [9], the authors of [64] describe the RWP mobility model as a stochastic process $\{V_i, R_i, S_i\}$, where V_i is the random variable denoting the velocity during trip i, R_i is the random variable denoting travel distance during trip i, and S_i is the random variable denoting travel time during trip i. Setting $\sum_{i=1}^{n} v_i(t)/n = V(t)$, then v_{RWP} can be expressed as follows:

$$v_{\mathrm{RWP}} = \lim_{t\to\infty} V(t) = \lim_{T\to\infty} \frac{1}{T}\int_{[0,T]} V(t)dt = \lim_{T\to\infty} \frac{\sum_{k=1,\ldots,K(T)} r_k}{\sum_{k=1,\ldots,K(T)} s_k} = \frac{E[R_i]}{E[S_i]}$$

where $K(T)$ is the total number of trips undertaken within time T, including the last one (possibly incomplete), and where r_k (respectively, s_k) is the travel distance (respectively, time) of trip k.

Thus, the computation of v_{RWP} is reduced to the problem of computing the expectation of the random variables, R_i and S_i. In [64], the authors show that

$$E[R_i] = \frac{2}{3}R_{\max} \text{ and } E[S_i] = \frac{2R_{\max}}{3(v_{\max} - v_{\min})} \cdot \ln\left(\frac{v_{\max}}{v_{\min}}\right)$$

yielding

$$v_{\text{RWP}} = \frac{v_{\max} - v_{\min}}{\ln\left(\frac{v_{\max}}{v_{\min}}\right)}$$

Furthermore, several interesting implications of the discussed characterization for v_{RWP} are presented in [64]. First, it is observed that $v_{\text{RWP}} \leq v_0$ and that $v_{\text{RWP}} = v_0$ if and only if $v_{\min} = v_{\max}$. This implies that the only way of avoiding speed decay is to avoid randomness in speed selection, imposing the same speed to a node during the entire simulation time. While having constant node velocity may be acceptable in some situations, the range of possible reference application scenarios for simulation is considerably reduced with this assumption. For instance, think about a scenario in which mobile nodes represent vehicles moving in a city: clearly, allowing vehicles to change speed during the travel (e.g., to reflect different speed limits) considerably increase simulation representativeness with respect to a situation in which the vehicle speed is fixed throughout the entire simulation time.

The authors of [64] observe that v_{RWP} becomes relatively closer to v_0 (thus reducing speed decay intensity and the time needed to reach stationary node velocity) as the speed range interval becomes smaller. A general recommendation to lessen speed decay is to shrink the allowed node speed interval, which comes at the price, however, of reducing the range of possible application scenarios for simulation.

A final and very interesting implication of the v_{RWP} characterization is that the *pdf* of the random variable S_i becomes heavy tailed when $v_{\min} \to 0$, implying that $E[S_i]$ becomes infinite, and $v_{\text{RWP}} \to 0$. Thus, if $v_{\min}$ is set to 0, the stationary regime of an RWP mobile network actually coincides with a static network ($v_{\text{RWP}} = 0$) and is reached only after infinite time. It is clear then that setting $v_{\min} = 0$, as it is actually very common in wireless network simulation, severely impacts simulation accuracy, since simulation results cannot be gathered before the node velocities have reached the stationary state.

6.4.3 The "Perfect" Simulation

In the previous sections, we have shown that how theoretical characterization of RWP mobile network properties can disclose sources of inaccuracy in wireless network simulation. Possible countermeasures have also been discussed, which essentially amounts to

(a) *simulation "warm-up"*: run the simulation for a relatively long time interval before starting collecting simulation results;
(b) *reducing speed range*: choose velocity from a smaller speed interval.

Unfortunately, both approaches for improving wireless network simulation approaches have considerable drawbacks, which discourage their usage in simulation practice. In particular, (a) causes considerable wastage of computational resources. Furthermore, estimating the time needed for the network to reach stationary conditions is difficult, and in some situations the time needed to reach stationarity can actually be infinite, for instance, when $v_{\min} = 0$. Moreover, the approach (b) also has considerable drawbacks, as it considerably reduces the range of possible reference application scenarios for simulation. Furthermore, (b) has effect only on the speed decay phenomenon, but cannot be used to mitigate the border effect.

Motivated by the above observations, researchers have made efforts to design a "perfect" simulation methodology, in which issues with simulation accuracy can be solved without incurring the drawbacks of approaches (a) and (b). A first noteworthy contribution in this direction is [65], where the authors present a methodology to remove the speed decay effect without reducing the speed range interval, with the only constraint that $v_{\min} > 0$. The authors' goal is to initialize the system directly in the stationary state, without the need of a "warm-up" period. The authors start deriving the *pdf* of the stationary average node velocity $\mathcal{V}_{\mathrm{RWP}}$ and show that $\mathcal{V}_{\mathrm{RWP}}$ cannot be directly used to initialize the system: if $\mathcal{V}_{\mathrm{RWP}}$ is used instead of a uniform distribution in $[v_{\min} > 0, v_{\max}]$ to select initial node velocities, the *pdf* of the resulting stationary average node velocity *changes*, and it is no longer $\mathcal{V}_{\mathrm{RWP}}$. Then, the authors show that a possible way of avoiding this problem is using a *composite* mobility model, where the *pdf* used to select initial node speed is different from that used to select the speed of next trips. In particular, the authors of [65] formally prove that the following methodology completely removes speed decay:

1. use $\mathcal{V}_{\mathrm{RWP}}$ to select speed of the first trip;
2. use default speed distribution (uniform in $[v_{\min} > 0, v_{\max}]$) to select speed of next trips.

In [43], the authors generalize the results of [65] to a wide class of mobility models (including RWP model, RD model) and show that the "perfect" simulation methodology defined in [65] can be used not only for average node speed but also for any structural network property admitting a stationary distribution. With respect to this, the authors of [43] show that a necessary and sufficient condition for a mobility model to admit stationary structural distributions is that the expected trip duration is finite. Thus, for models such as RWP, the "perfect" simulation methodology of [65] can be used to remove not only the speed decay but also the border effect.

6.5 Formal Studies of Connectivity on MANETs' Models

6.5.1 Connectivity Threshold for Mobility Models

As described in the previous section, the border effect may considerably impact simulation accuracy. In this section, we analyze the consequence of the border effect on the formal analysis of properties for MANETs, in particular referring to the critical transmission range for connectivity.

- *Connectivity threshold for mobile models*. Using the previous result, Santi [51] studies the connectivity threshold for mobile networks. His model is the following: There are n vertices deployed uniformly at random in the unit square $[0, 1]^2$. The nodes move randomly, but the mobility model is not fixed, it only must meet two conditions: it must be *bounded* and *obstacle free*. A mobility model $\mathcal{M}$ is said to be *bounded* if the support of the probability density function *pdf* of the long-term distribution of the nodes is contained in $[0, 1]^2$. Similarly, $\mathcal{M}$ is said to be *obstacle free* if the support of the *pdf* contains $[0, 1]^2 \setminus \partial[0, 1]^2$, where $\partial[0, 1]^2$ denotes the boundary. In other words, every subregion with non-zero measure has to have positive probability to contain at least one node at a given time. Notice that the random direction model, the random waypoint model, and Brownian motion are all bounded and obstacle free. Moreover, not necessarily all nodes have to move at the same speed, each one can choose its speed from an interval $[v_{\min}, v_{\max}]$. Also, the nodes can pause for a predefined amount of time t_p after having reached their destination.

In particular, due to border effects and due to different node velocities, the long-term spatial distribution of the nodes might be different from the starting distribution, even if they start with the uniform distribution. Define the *mobile threshold for connectivity* $r_{\mathcal{M}}$ as the minimum value of the radius r, such that when taking a snapshot of the graph chosen from the long-term spatial distribution of the nodes, the graph is connected. Notice $r_{\mathcal{M}}$ might be different from the threshold of the static case $r_c = \sqrt{\frac{\log n}{\pi n}}$. In fact, the first result of the paper states that if the *pdf* of the mobility model $f_{\mathcal{M}}$ is continuous on $\partial[0, 1]^2$ and $\min f_{\mathcal{M}} > 0$, then a.a.s. $r_{\mathcal{M}} = c\sqrt{\frac{\log n}{\pi n}}$ with $c \geq 1$. The proof uses the fact that in the static case, a.a.s. the threshold of connectivity equals the longest edge of the Euclidean minimum spanning tree built on the n points (see [45]).

The second result the paper states that in the random waypoint model with pause time t_p and $v = v_{\min} = v_{\max}$, a.a.s. the connectivity threshold of the long-term spatial distribution $r^w_{t_p} = \frac{t_p + \frac{0.521405}{v}}{t_p}\sqrt{\frac{\log n}{\pi n}}$, for $t_p > 0$, and $r_0 \gg \frac{\log n}{n}$, for $t_p = 0$. Intuitively the results says that when nodes stop at the waypoint for a positive amount of time before choosing the next waypoint, the connectivity threshold of the long-term distribution differs from the static case by only a constant factor. In the case when $t_p \to \infty$, $r^w_{t_p} \to r_c$, and the long-term spatial distribution becomes the uniform distribution. On the other hand, if the nodes start traveling toward the next waypoint immediately after touching the current waypoint, the connectivity threshold is asymptotically larger than in the static case. The intuition behind this result is as follows: the formula for the *pdf* contains two components; one for the time a node is resting at a waypoint, which is uniform since the waypoint is chosen uniformly at random, and a mobility component responsible for border effects. If the uniform component of the *pdf* is not 0, it asymptotically dominates over the mobility component, and the connectivity threshold is asymptotically the same as in the static case. On the contrary, if the uniform component is 0, the *pdf* coincides with the mobility component, which has a different asymptotic behavior than uniform, implying a larger connectivity threshold.

6.5.2 Connectivity Periods on Mobile Models

- *The walkers' model on the grid.* The authors in [18] present a model of establishment and maintenance of communication between mobile nodes, denoted *walkers* in the paper, where the nodes move in a fixed environment modeled by a toroidal grid T. Therefore, the authors present a hybrid random model. The model is defined as follows: given a toroidal square grid in the plane $T = (V, E)$ with $|V| = N = n^2$, a set W of *walkers* with $|W| = w$, and a "transmitting distance" d (the same for all the walkers), the w walkers are sprinkled randomly and independently on the N vertices of T (a vertex may contain more than one walker). Two walkers w_1 and w_2 can communicate in one hop if the Euclidean distance between the position of the walkers is at most d. Two walkers can communicate if they can reach each other by a sequence of such hops.

Then, in a synchronized way, each walker performs an independent standard random walk on the nodes of T. That is, each walker moves at each time step to one of the four neighboring vertices, all chosen with equal probability $1/4$. Hence, for any time $t \in \mathbb{N}$, one can define the *random graph of walkers* $\mathcal{W}_t(T, w, d)$: the vertices of this graph are the w walkers together with their position they are occupying on T at time t, and there is an edge between two walkers if their Euclidean distance is at most d (if more than one walker occupies a vertex of the grid, the authors do not consider the corresponding multigraph and consider that position of the grid as if there was only one walker). The authors then study the behavior (as $N \to \infty$) of the connectivity and disconnectivity of $\mathcal{W}_t(T, w, d)$ for any $t \in \mathbb{N}$, where $\mathcal{W}_0(T, w, d)$ is formed by the initial distribution of the walkers on T (see Fig. 6.3 for a toy example of one step).

The paper first examines the initial *static* case $\mathcal{W}_0(T, w, d)$, which is a snapshot of the process at one point in time: in particular, the paper studies the distribution of the number of isolated vertices of $\mathcal{W}_0(T, w, d)$ as well as some other information which helps to answer the dynamic questions. Define h to be the number of grid points within distance d of any fixed point in T. Clearly, $h = \Theta(d^2)$. If $d = \Omega(n)$, then $\mathcal{W}_0(T, w, d)$ is connected a.a.s., so the interesting case is $d = o(n)$, i.e., $h = o(N)$. Furthermore denote by $\rho = w/N$ be the expected number of walkers at a vertex and define the parameter $\mu = N\left(1 - e^{-\rho}\right)e^{-h\rho}$. The authors first prove that in the static initial case at time $t = 0$, $\mathbf{Pr}\left[\mathcal{W}_0(T, w, d) \text{ is connected}\right] = e^{-\mu} + o(1)$.

Using the information from the static case, in the dynamic setting, the crux of the paper is the study, as t evolves, of the birth and death of isolated vertices, and the sudden connection and disconnection of $\mathcal{W}_t(T, w, d)$. Let LD_t be the random variable counting the length of the disconnected period (similarly, a random variable

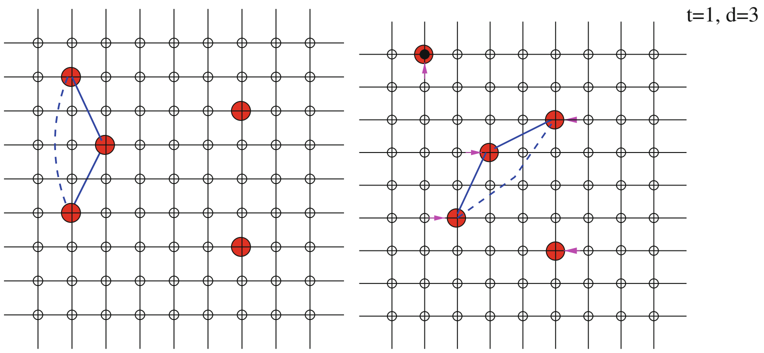


Fig. 6.3 A step of the walkers' problem on the grid. The *solid line* represents direct communications of the ad hoc network, the *dashed line* represents communication between nodes that are at distance more than d

LC_t counting the length of the connected period is considered) starting at time step t provided that it really starts to be disconnected at t. Define the *average length* of a disconnected period starting at time t to be $\mathrm{LD}_{\mathrm{av}} := \mathbf{E}\left(\mathrm{LD}_t \mid \mathrm{LD}_t > 0\right)$, which is independent of t, and is a function of N, d, and w. The authors show that the following hold about $\mathrm{LD}_{\mathrm{av}}$:

$$\mathrm{LD}_{\mathrm{av}} \sim \begin{cases} \frac{e^{\mu}-1}{\mu b \rho} & \text{if } d\rho \to 0 \\ \frac{e^{\mu}-1}{1-e^{-\lambda}} & \text{if } d\rho \to c \\ e^{\mu} & \text{if } d\rho \to \infty \end{cases}$$

where $b = \Theta(d)$ is a function related to the boundary of the ball of radius r in T and $\lambda = \left(1 - e^{-b\rho}\right)\mu$ with $0 < \lambda < \mu$ for $d\rho \to c$. Furthermore, LD_t converges in probability for $t \to \infty$ (N fixed) to a random variable LD, where $\mathrm{LD} \sim \mathrm{LD}_{\mathrm{av}}$ a.a.s. Similar results can be given for the average length of connected periods. For the proof, the authors calculate joint factorial moments of variables accounting for births, deaths, and survivals of isolated vertices, and they show that the connectivity (disconnectivity, respectively) of the graph is asymptotically equivalent to the non-existence (existence, respectively) of isolated vertices.

The results in the paper are proved in full generality, under any norm and for $T = [0, 1)^m$ for $m = \Theta(1)$. Also, the paper proves results on the connectivity and disconnectivity periods for the case when the underlying graph of fixed paths is a cycle.

- *The DRGG model with radii* r_c. The paper [15] studies the connectivity of a random direction type model for MANETs. The model is a RGG at the connectivity threshold r_c, where all vertices move at the same speed. This dynamic model is denoted by the authors as the dynamic random geometric graph. More formally, the model is the following: at the starting of the process ($t = 0$), n nodes are scattered independently and uniformly at random in the unit torus $[0, 1)^2$. At any time $t \in \{0, 1, 2, 3, \ldots\}$, two nodes are connected if their Euclidean distance is at most r. The authors fix the value of r to be the value at the connectivity threshold for static RGG, i.e., $r = r_c = \sqrt{\frac{\log n \pm O(1)}{\pi n}}$. The dynamic model is the following: given two positive reals $s = s(n)$ and $m = m(n)$, at any time step t, each node i jumps a distance s in some direction $\alpha_{i,t} \in [0, 2\pi)$. The initial angle $\alpha_{i,0}$ is chosen independently and uniformly at random for each node i and then at each time step each node changes its angle independently with probability $1/m$. Thus, the number of steps a node has to wait before changing its direction follows a geometric distribution with expectation m. New angles are also selected independently and uniformly at random in $[0, 2\pi)$ (see Fig. 6.4 for a toy example of the changes of the graph in a single step).

The goal of the paper is to analyze the expected length of (dis)connectivity periods of the underlying graph. To state the main result more formally, denote by $\mathcal{C}_t$ the event that the random graph is connected at time t and similarly denote by $\mathcal{D}_t$ the event that the graph is disconnected at time t. Furthermore, define $L_t(\mathcal{C})$ to be the random variable counting the number of consecutive steps that $\mathcal{C}$ holds starting from time t (possibly ∞ and also 0 if $\mathcal{C}_t$ does not hold). $L_t(\mathcal{D})$ is defined analogously by interchanging $\mathcal{C}$ with $\mathcal{D}$. It can be shown that the distribution of $L_t(\mathcal{C})$ and $L_t(\mathcal{D})$ is independent of t. Define also

$$\lambda_{\mathcal{C}} = \mathbf{E}\left(L_t(\mathcal{C}) \mid \mathcal{D}_{t-1} \wedge \mathcal{C}_t\right) \quad \text{and} \quad \lambda_{\mathcal{D}} = \mathbf{E}\left(L_t(\mathcal{D}) \mid \mathcal{C}_{t-1} \wedge \mathcal{D}_t\right)$$

that is, $\lambda_{\mathcal{C}}$ ($\lambda_{\mathcal{D}}$, respectively) count the expected number of steps that the graph stays connected (disconnected, respectively) starting at time t conditional upon the fact that it becomes connected (disconnected) precisely at time t. The main result of the paper is the following: if $srn = \Theta(1)$, then

$$\lambda_{\mathcal{C}} \sim \frac{1}{1 - e^{-\mu(1-e^{-4srn/\pi})}} \quad \text{and} \quad \lambda_{\mathcal{D}} \sim \frac{e^{\mu} - 1}{1 - e^{-\mu(1-e^{-4srn/\pi})}}$$

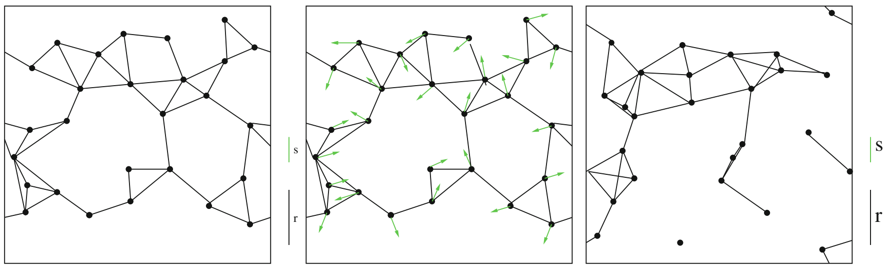


Fig. 6.4 A step in the DRGG. Starting at a given ad hoc graph (*left picture*), every node chooses a new direction chosen at random (*center picture*), creating a new ad hoc graph (*right picture*)

Otherwise, it is

$$\lambda_{\mathcal{C}} \sim \begin{cases} \frac{\pi}{4\mu srn} & \text{if } srn = o(1) \\ \frac{1}{1-e^{-\mu}} & \text{if } srn = \omega(1) \end{cases} \quad \text{and} \quad \lambda_{\mathcal{D}} \sim \begin{cases} \frac{\pi(e^{\mu}-1)}{4\mu srn} & \text{if } srn = o(1) \\ e^{\mu} & \text{if } srn = \omega(1) \end{cases}$$

One can observe that for $srn = o(1)$ and $srn = \omega(1)$ the results of $\lambda_{\mathcal{C}}$ and $\lambda_{\mathcal{D}}$ correspond to the respective limits in the case when $srn = \Theta(1)$. These results have various consequences; on the one hand the expected number of steps in a period of connectivity (disconnectivity) does not depend on m, that is, it does not depend on how often the nodes change their direction. On the other hand, $\lambda_{\mathcal{C}}$ and $\lambda_{\mathcal{D}}$ are non-decreasing in s. The intuition behind this is as follows: if the distances between two time steps are big, the correlations between two consecutive steps are smaller, and connectivity/disconnectivity changes more frequently. For a very large s (case $srn = \omega(1)$), $\lambda_{\mathcal{C}}$ and $\lambda_{\mathcal{D}}$ do not depend on s anymore, since for such a value of s two consecutive steps are roughly independent. Finally, one can observe that in the case $srn = o(1)$ models the underlying continuous-time model very well: denote by $\tau_{\mathcal{C}} = s\lambda_{\mathcal{C}}$ ($\tau_{\mathcal{D}} = s\lambda_{\mathcal{D}}$, respectively) the distance covered by each vertex during a connectivity (disconnectivity) period. Then,

$$\tau_{\mathcal{C}} \sim \frac{\pi}{4\mu rn} \sim \frac{\pi\sqrt{\pi}}{4\mu\sqrt{n \ln n}}, \qquad \tau_{\mathcal{D}} \sim \frac{\pi(e^{\mu}-1)}{4\mu rn} \sim \frac{\pi\sqrt{\pi}(e^{\mu}-1)}{4\mu\sqrt{n \ln n}}$$

which asymptotically do not depend on s. Since these results also hold if s tends to 0 arbitrarily fast, the related continuous-time model has a similar behavior: in that model the traveled distance during periods of connectivity (disconnectivity) also does not depend on the average distance sm between changes of angle.

The main ingredient of the proof is the fact that the probabilities needed to compute $\lambda_{\mathcal{C}}$ and $\lambda_{\mathcal{D}}$ can be expressed in terms of the probabilities of events involving only two consecutive steps. This is surprising, since in this case (in contrast to the article [18]) the sequence of connected/disconnected states is not Markovian—staying connected for a long period of time makes it more likely to remain connected for one more step. As in the article [18], it turns out that the existence/non-existence of isolated vertices is asymptotically equivalent to the disconnectivity/connectivity of the graph, both in the static case and for two consecutive steps. Although the proof is technically very different from the one in [18], it is similar in spirit: the characterization of the changes of the number of isolated vertices between two consecutive steps is based on the computation of the joint factorial moments of the variables accounting for these changes (births/deaths/survivals of isolated vertices). As in [18], it is not obvious that the probability of existence of components of larger sizes in the dynamic model is negligible compared to the probability of sudden appearance of isolated vertices, but in the paper it is shown to be the case.

6.5.3 The Effect of Mobility to Speed up Message Dissemination in Sparse Networks

In this section we survey in chronological order three results which show that high mobility of nodes helps in disseminating information.

- *The Source–destination pairs model.* The work [27] can be considered as the first attempt to formally analyze a model of mobility. The model is the following: there are n nodes ($n \to \infty$) all lying in the disk of unit area. The location of the ith node at time t is given by the random variable $X_i(t)$. Each of the n nodes is a source node for one session and a destination node for another session, and each node i has an infinite stream of packets to send to its destination $d(i)$. The source–destination (S–D) association is established initially and does not change over time. The nodes are mobile, but the mobility model described by the authors is non-constructive: the process $\{X_i(\cdot)\}$ is stationary and ergodic with stationary distribution uniform on the disk, and trajectories of different nodes are independent and identically distributed. It is a drawback of the paper, that the exact movement of the nodes is not explained: in particular, it is not clear what happens when a node touches the boundary of the disk. Recall that as mentioned before, boundary effects can change the distribution. The information exchange is not restricted to nodes within a certain distance, but it is the following: at slotted time t, node i has transmission power $P_i(t)$. Denote by $\gamma_{ij}(t)$ the channel gain from node i to node j, such that the received power at node j is $P_i(t)\gamma_{ij}(t)$. Formally, $\gamma_{ij}(t)$ is defined as $\frac{1}{|X_i(t)-X_j(t)|^\alpha}$, where α is a parameter greater than 2. Node i can transmit to node j if

$$\frac{P_i(t)\gamma_{ij}(t)}{N_0 + \frac{1}{L}\sum_{k\neq i} P_k(t)\gamma_{kj}(t)} > \beta \tag{6.2}$$

  where β is the signal-to-interference ratio requirement for successful communication, N_0 is the background noise power, and L is the processing gain of the system, it can be taken to be 1. Intuitively speaking, on the one hand, the closer j to i at time t, the bigger $\gamma_{ij}(t)$, and the more likely it is that node i can transmit a packet to node j. On the other hand, relative distances between nodes also play a role: if a node i is close to neighbor j, but j has many other neighbors very close, and at the same time i is further away from another node j', whose neighbors are all further away than i, it might happen that i is able to transmit to j' and not to j. In the following it is assumed that all nodes transmit at the same power P. Whether or not a node transmits to another one is decided by an external scheduler. Every node is assumed to have an infinite buffer to store packets, and when packets are transmitted from source to destination, they can go through one or more other nodes serving as relays. The goal is to find a scheduling policy with high long-term throughput. To make this concept more precise, define by $M_i^\pi(t)$ the number of source node i packets that $d(i)$ receives at time t under the scheduling policy π. A throughput $\lambda(n)$ is feasible, if there exists a policy π such that for *every* S–D pair i we have

$$\lim_{T\to\infty} \infty \sum_{t=1}^{T} M_i^{\pi}(t) \geq \lambda(n)$$

and the goal is to maximize $\lambda(n)$.

The authors first prove a lower bound in a dynamic model where relay nodes are forbidden. More precisely, they show that there exists a constant $c > 0$ such that the probability of having a throughput of at least $cn^{-(1/(1+\alpha/2))}$ tends to 0 for n sufficiently large. The theorem is stronger if α is closer to 2: if $\alpha \to 2$, the probability of a throughput of $c/\sqrt{n}$ tends to 0. This is the same lower bound as in the static model [23]. The intuition behind this result is the following: if long distances are allowed, then interference limits the number of concurrent transmissions. If a scheduling policy allows only short transmissions, then only a small fraction of S–D pairs is sufficiently close to transmit a packet.

Next, as a main result of their paper, the authors show that mobility helps if intermediate relay nodes are permitted. If for every S–D pair every other node can serve as intermediate relay (that is, at different time slots different nodes may contain part of the packet stream between i and $d(i)$), an asymptotically optimal throughput of $\lambda(n) = c$ for some $c > 0$ can be attained. To prove this the authors consider the following scheduling policy: every packet is relayed at most once. For every time slot t, the set of nodes is randomly partitioned into a set of potential senders (of size sn for some constant $s > 0$) and potential receivers. Each sender node may transmit packets to its nearest neighbor among all receiver nodes, and the sender indeed transmits if the interference generated by other senders is sufficiently small (according to the formula given in (6.2)). The algorithm runs in two interleaved phases: in phase 1 (in odd time slots, say) packets are sent only from source nodes to relays (or directly to the destination node), in phase 2 (in even time slots, say) packets are sent only from relays to destination nodes. The proof of the result uses the fact that at any particular moment in time the distribution of the points is uniform on the disk, together with some results on the asymptotic distribution of extrema of i.i.d. random variables. We recall once again, that is not clear how the nodes move and what happens when touching the boundary.

- *The DRGG model below* r_t. In the work [38] the authors study a very general random direction-type model with a radius below the threshold of the existence of a giant component. More precisely, the authors consider the following model: at the beginning n nodes ($n \to \infty$) are distributed uniformly at random in a square $\mathcal{A} = \mathcal{L} \times \mathcal{L}$, where $L = c\sqrt{n}$ for some large constant $c > 0$. Two nodes can exchange information if they are within Euclidean distance 1. It is assumed that information exchange takes zero time, once two nodes are at distance ≤ 1. By the choice of L, $n/\mathcal{A}$ tends to a small constant ($n/\mathcal{A} < 1/\pi$), which in the static case corresponds to a random geometric graph below the thermodynamical limit $r_t = c/\sqrt{n}$. Recall in Sect. 6.2 we already pointed that for a radius r below the thermodynamical limit r_t, the RGG is disconnected and it does not have yet a giant component. The mobility model is the following: the nodes follow random trajectories with Poisson rate τ, keeping uniform speed between direction changes. When a node hits the boundary at an incidence angle θ, it follows the mirror reflection policy, i.e., the node bounces back at angle $\pi - \theta$. Therefore, the probability density for a node to travel a time t in a certain direction before changing the direction is

$$\frac{1}{2\pi}\tau \exp(-\tau t)$$

where τ is a parameter controlling the speed of change. Notice that if $\tau \to \infty$ then the mobility represents Brownian motion, while if $\tau \to 0$ the mobility represents a random waypoint model with the mirror reflection policy, where the nodes only change direction when touching the boundary of the square. The factor $\frac{1}{2\pi}$ comes from the fact that every angle has the same probability to be chosen.

The authors give an upper bound on the speed at which information can be propagated between any pair of nodes. Recall that in the static case information between most pairs of nodes cannot be propagated since the largest connected component for the value of $\nu := n/\mathcal{A}$ to be considered has size $O(\log n)$. The authors show that mobility helps to propagate information. In order to state the result more precisely, consider a node that starts at coordinate $z_0 = (x_0, y_0)$ at time $t = 0$ that wants to propagate information to a destination node starting at coordinate $z_1 = (x_1, y_1)$. The authors show that the destination node can be assumed to be fixed without changing the asymptotic results of the analysis. Denote by $q_\nu(z_0, z_1, t)$ the probability that the destination receives the information before time t (n is assumed to be large, but the density ν is a constant). A scalar $s_0 > 0$ is called an upper bound for the propagation speed, if for all $s > s_0$, $\lim q_\nu\left(z_0, z_1, \frac{|z_0 - z_1|}{s}\right) = 0$ whenever $|z_1 - z_0| \to \infty$. Using this definition, the authors show that an upper bound on the information propagation speed is

$$\min_{\rho,\Theta>0}\left\{\frac{\Theta}{\rho} \text{ with } \Theta = \sqrt{\rho^2 v^2 + \left(\tau + \frac{\frac{n}{\mathcal{A}}4\pi v I_0(\rho)}{1 - \frac{n}{\mathcal{A}}\pi\frac{2}{\rho}I_1(\rho)}\right)^2} - \tau\right\} \tag{6.3}$$

where v is the maximum node speed, $I_0()$ and $I_1()$ are *modified Bessel functions* defined by $I_0(x) = \sum_{k\geq 0}\left(\frac{x}{2}\right)^{2k}\frac{1}{(k!)^2}$, and $I_1(x) = \sum_{k\geq 0}\left(\frac{x}{2}\right)^{2k+1}\frac{1}{(k+1)!k!}$. To get some intuition about this bound and its involved parameters, note that the quantities $I_0(x)$ and $\frac{2}{x}I_1(x)$ are both larger than 1, and therefore the expression has meaning if $\frac{n}{\mathcal{A}} < \frac{1}{\pi}$, as above the thermodynamical limit there is a giant component, and therefore the information propagation speed is infinity. Observe also that the obtained value is larger if τ is larger. Such a behavior is expected, since changing directions more frequently may result in faster information propagation and therefore the propagation speed might be higher. Finally, ρ and Θ are parameters that correspond to the Laplace transform of the sequence of nodes such that a piece of information is visiting on its way from source to destination (see below for a rough explanation).

To prove the result (6.3), the authors decompose the journey (which is the sequence of nodes a piece of information undergoes from the source to the destination) into different segments. These segments either correspond to node movements through which the information is propagated or to direct propagations between two nodes, when a node immediately, without movement, propagates the information to another one due to the fact that the two nodes are at distance ≤ 1. The authors consider the segments as independent, which is not true, since, for example, two consecutive nodes in the sequence are more likely to move in opposite directions or node speeds are different, and a faster moving node meets more nodes, but they show that in this way they prove an upper bound on the propagation speed for the real model, and hence the assumption is justified.

On the technical side, the authors compute the Laplace transform of the probability density of a fixed journey of length k, defined as a journey where $k+1$ nodes participate in the process of information propagation from the source node to the destination node. Since the segments are considered to be independent, the Laplace transform of the journey is the product of the Laplace transform of the segments. In particular, the Laplace transform of such a journey does not depend on the particular nodes participating, but only on the length of the journey. As the journey, however, is not known in advance, the authors consider the Poisson generating function $G(Z, (\rho, \Theta))$ whose nth coefficient is the Laplace transform of all journeys in a network with n nodes in a square of size $\mathcal{A}$. They show that this generating function is equivalent to an ordinary generating function whose kth coefficient is the Laplace transform of the probability density of a fixed journey of length k. Hence, for $n \to \infty$ an upper bound for the asymptotic behavior of $q_n(z_0, z_1, t)$ can be calculated from simpler expressions for journeys composed of independent segments. The asymptotic growth of the Laplace transform of $q_\nu(z_0, z_1, t)$ is then obtained by those values of (ρ, Θ) for which the denominator corresponding to the nth coefficient of the Poisson generating function $G(\nu, (\rho, \Theta))$ vanishes. The final expression for $q_\nu(z_0, z_1, t)$ is then obtained using the inverse Laplace transform.

One has to point out that the conference version of the article, although sounding very plausible, is not easy to read. In particular, the probability spaces are not clearly defined.

- *The hybrid grid model approximating DRGG, for $r > r_t$.* In the Chapter, *Information Spreading in Dynamic Networks: An Analytical Approach*, Andrea Clementi and Francesco Pasquale give an extensive presentation of this model and other previous related models in the specific framework of information spreading in dynamic networks. However, for completeness of our survey, we also briefly sketch the model. We refer the reader to the mentioned chapter in the present book. In the model used by [13] a RGG is approximated by a very fine grid on which the nodes are restricted to move. Hence, it is a discretized version (with respect to both time and space) of the models used in [38]: there are n nodes ($n \to \infty$) moving on the corner points of a grid inside a square of size $\sqrt{n}$. In more detail, for some given $\varepsilon > 0$, at any time t the nodes occupy one position of $L(n, \varepsilon)$, where

$$L(n, \varepsilon) = \left\{ (i\varepsilon, j\varepsilon) \mid i, j \in \mathbb{N} \wedge i, j \leq \frac{\sqrt{n}}{\varepsilon} \right\}$$

The position at time $t = 0$ is chosen uniformly at random, independently for all nodes, and at any fixed time slot t two nodes are connected by an edge if their Euclidean distance is less than r. Here $r \geq r_0$, where r_0 is a sufficiently large constant. Therefore, the graph contains a giant component, but is not necessarily connected a.a.s., which would happen only for $r \geq c \log n$. The mobility model is the following: for a given *move radius* ρ, define the *move graph* $M_{n,\rho,\varepsilon} = (L_{n,\varepsilon}, E_{n,\rho,\varepsilon})$, where

$$E_{n,\rho,\varepsilon} = \{(p, q) \mid p, q \in L_{n,\varepsilon},\ (p, q) \leq \rho\}$$

and $d(\cdot, \cdot)$ is the Euclidean distance. Furthermore, for any position p in the square, define by $\Gamma(p) = \{q \mid (p, q) \in E_{n,\rho,\varepsilon}\}$. A node at position p at time t chooses uniformly at random its position at time $t + 1$ among all elements of $\Gamma(p)$. In other words, it chooses a random node in a ρ-vicinity of the original position (see Fig. 6.5 for toy example of one step in the present model). Initially, at time $t = 0$, one node, the source node, contains a message that should be broadcast to every other node of the network. Whenever at a certain time slot t one node u contains the message and there is another node v within distance r that does not yet contain it, the message is broadcast from u to v. It is assumed that transmission takes zero time. Recall that

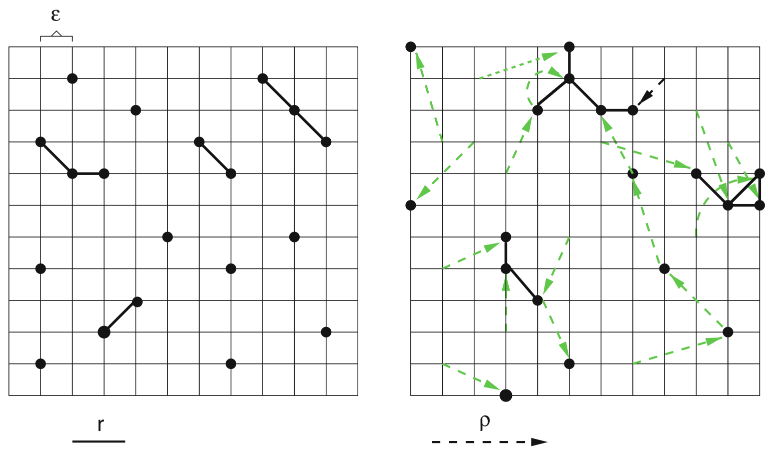


Fig. 6.5 Two consecutive time steps in the model of [13]. On the *left* the graphs at some fixed time t, where a node connected with all the other nodes at distance $\leq r$. *Right picture*: the resulting graph after a movement of each vertex of a distance $\leq \rho$. The trajectory of movement is indicated by the *light dotted* arrows

the flooding time is the number of time steps required to broadcast the message to all nodes in the network.

The authors prove the following: if $\rho \geq c \log n$ for some constant $c > 0$, then the flooding is a.a.s. completed after

$$O\left(\frac{\sqrt{n}}{\rho} + \log n\right)$$

time steps, which is asymptotically almost tight since the expected flooding time is $\Omega\left(\sqrt{n}/\rho\right)$. That is, if the move radius is sufficiently large (i.e., the node velocity is sufficiently high), the flooding time is independent of r (as long as $r \geq r_0$). This is especially interesting for r below the connectivity threshold: flooding can be completed although at every time step the graph is disconnected.

The proof of the result uses a tessellation argument; the square is subdivided into supercells of side length $\Theta(\rho)$. The proof proceeds in the following three steps: first, it is shown that after $O(\log n)$ time steps there is a.a.s. at least one supercell which contains $\Theta(\rho^2)$ informed nodes (the supercell is called quasi-informed). Next, in a second phase, it is shown that, with high-probability, any quasi-informed supercell at time t makes all its adjacent supercells quasi-informed at time $t + 1$. Since any supercell set D has a boundary of size at least $\Theta\left(\sqrt{|D|}\right)$, after $O\left(\sqrt{n}/\rho\right)$ time steps all supercells are quasi-informed a.a.s. Finally, in a last phase, it is shown that in $O(\log n)$ time steps a.a.s., any quasi-informed cell becomes completely informed. That is, all nodes of that cell contain the message that should be broadcast.

6.6 Conclusions

We surveyed the main theoretical issues when studying models for MANETs. We described some of the models, where properties have been investigated with a certain degree of formal rigor.

In particular, in Sect. 6.4 we have presented theoretical characterizations of fundamental properties such as node spatial distribution and average velocity, under the assumption that nodes move according to the RWP mobility model. In the same section, we have shown how such characterizations have been used to disclose accuracy issues with wireless network simulation practice and to design a “perfect” simulation methodology solving these issues.

In Sect. 6.5 we presented recent papers dealing with connectivity issues of dynamical models, where nodes move synchronously on $[0, 1)^2$. The goal in [51] is to study how mobility affects the threshold of connectivity. The author gives the threshold under certain conditions affecting mobility parameters. The papers [18] and [15] compute the expected lengths of connectivity and disconnectivity periods of vertices that are moving on a predetermined grid (in the case of [18]) and of vertices of a dynamic geometric graph whose radius is at the threshold of connectivity (in the case of [15]). The remaining three papers deal with the issue of how mobility can be used to maintain the transmission range small while at the same time allowing for connectivity properties. The papers of [38] and [13] are complementary: whereas the authors in [38] study random geometric graphs with a radius below the thermodynamical threshold, the paper [13] considers the case of radii between the thermodynamical threshold and the threshold of connectivity. The third paper studied here, the work of [27], is orthogonal to these two since there is no absolute bound on the radius of transmission, but it also supports the hypothesis that mobility can help in propagating information.

Acknowledgments This research was partially supported by the FP7-ICT-21527 project of the EC, FRONTS. The authors are grateful to the *Centre de Recerca Matemàtica*, Universitat Autònoma de Barcelona for hospitality and support.

References

1. V. Anand A. Bharathidasas. *Sensor Networks: An Overview*. *Technical Report*, Department of Computer Science, University of California at Davis, 2002.
2. C. Avin and G. Ercal. On the cover time and mixing time of random geometric graphs. *Theoretical Computer Science*, 380(1-2):2–22, 2007.
3. P. K. Agarwal, L. J. Guibas, H. Edelsbrunner, J. Erickson, M. Isard, S. Har-Peled, J. Hershberger, C. Jensen, L. Kavraki, P. Koehl, M. Lin, D. Manocha, D. Metaxas, B. Mirtich, D. Mount, S. Muthukrishnan, D. Pai, E. Sacks, J. Snoeyink, S. Suri, and O. Wolefson. Algorithmic issues in modeling motion. *ACM Computing Surveys*, 34(4):550–572, 2002.
4. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38:393–422, 2002.
5. A. Boukerche and L. Bononi. Simulation and modeling of wireless, mobile, and ad hoc networks. In S. Basagni, M. Conti, S. Giordano, and I. Stojmenović, editors, *Mobile Ad Hoc Networking*, chapter 14, pages 373–409. IEEE Press, New York, NY, 2004.
6. C. Bettstetter. Mobility modeling in wireless networks: categorization, smooth movement, and border effects. *Mobile Computing and Communications Review*, 5(3):55–66, 2001.
7. C. Bettstetter. Smooth is better than sharp: a random mobility model for simulation of wireless networks. In *Proceedings of the 4th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 19–27, ACM, New York, 2001.
8. H. Breu and D. G. Kirkpatrick. Unit graph recognition is NP-hard. *Computational Geometry*, 9:3–24, 1998.
9. C. Bettstetter, G. Resta, and P. Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(3):257–269, 2003.
10. T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2(5):483–502, 2002.
11. Y. A. Chau and Y.-H. Chen. Analytical link lifetime of a manet based on the three-dimensional brownian mobility model. In *Proceedings of the International Symposium on intelligent Signal Processing and Communication Systems, (ISPACS)*, pages 428–431, IEEE Computer Society, Los Alamitos, CA, 2007.
12. B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990.
13. A. E. F. Clementi, F. Pasquale, and R. Silvestri. Manets: High mobility can make up for low transmission power. In S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. E. Nikoletseas, and W. Thomas, editors, *Proceedings of 36th International Collogquium on Automata, Languages and Programming*, pages 387–398, 2009.

14. C. de Morais Cordero and D. P. Agrawal. *Ad Hoc and Sensor Networks*. World Scientific, New Jersey, 2006.
15. J. Díaz, D. Mitsche, and X. Pérez. Connectivity for dynamic random geometric graphs. *IEEE Transactions on Mobile Computing*, 8:821–835, 2009.
16. J. Díaz, D. Mitsche, and X. Pérez. On the probability of existence of mid-size components in random geometric graphs. *Advances of Applied Probability*, 41:1–14, 2009.
17. J. Díaz, J. Petit, and M. J. Serna. A random graph model for optical networks of sensors. *IEEE Transactions on Mobile Computing*, 2:143–154, 2003.
18. J. Díaz, X. Pérez, M. J. Serna, and N. Wormald. On the walkers problem. *SIAM Journal of Discrete Mathematics*, 22:747–775, 2008.
19. S. Dolev, E. Schiller, and J. L. Welch. Random walk for self-stabilizing group communication in ad hoc networks. *IEEE Trans. Mobile Computing*, 5(7):893–905, 2006.
20. L. J. Guibas, J. Hershberger, S. Suri, and L. Zhang. Kinetic connectivity for unit disks. *Discrete & Computational Geometry*, 25:591–610, 2001.
21. E. N. Gilbert. Random plane networks. *Journal of the Society for Industrial and Applied Mathematics*, 9:533–543, 1961.
22. P. Gupta and P. R. Kumar. Critical power for asymptotic connectivity in wireless networks. In W. McEneaney, G. G. Yin, and Q. Zhang, editors, *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming*, pages 547–566. Birkhäuser, 1998.
23. P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46:388–404, 2000.
24. M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, NY, 1980.
25. A. Goel, S. Rai, and B. Krishnamachari. Sharp thresholds for monotone properties in random geometric graphs. *Annals of Applied Probability*, 15:364–370, 2005.
26. G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, 2001.
27. M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Transactions on Networking*, 10(4):477–486, 2002.
28. R. A. Guerin. Channel occupancy time distribution in a cellular radio system. *IEEE Transactions on Vehicular Technology*, 36(3):89–99, 1987.
29. S. Guo and O. W. W. Yang. Energy-aware multicasting in wireless ad hoc networks: A survey and discussion. *Computer Communications*, 30:2129–2148, 2007.
30. P. G. Hall. *Introduction to the Theory of Coverage Processes*. Wiley, New York, NY, 1988.
31. R. Hekmat. *Ad-hoc Networks: Fundamental Properties and Network Topologies*. Springer, Heidelberg, 2006.
32. X. Hong, M. Gerla, G. Pei, and C.-C. Chiang. A group mobility model for ad hoc wireless networks. In *Proceedings of the 2nd ACM international workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 53–60, ACM, New York, NY, 1999.
33. E. Hyytiä, P. Lassila, and J. Virtamo. Spatial node distribution of the random waypoint mobility model with applications. *IEEE Transactions on Mobile Computing*, 5(6):680–694, 2006.
34. M. Hollick, I. Martinovic, T. Krop, and I. Rimac. A survey on dependable routing in sensor networks, ad hoc networks, and cellular networks. In *EUROMICRO Conference*, pages 495–502, Los Alamitos, CA, 2004. IEEE Computer Society.
35. Z. J. Haas and M. R. Pearlman. The performance of query control schemes for the zone routing protocol. *IEEE/ACM Transactions on Networking*, 9(4):427–438, 2001.
36. A. P. Jardosh, E. M. Belding-Royer, K. C. Almeroth, and S. Suri. Real-world environment models for mobile network evaluation. *IEEE Journal on Selected Areas in Communications*, 23(3):622–632, 2005.
37. D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

38. P. Jacquet, B. Mans, and G. Rodolakis. Information propagation speed in mobile and delay tolerant networks. In *Proceeding of the 29th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Los Alamitos, CA, 2009. IEEE Computer Society.
39. L. M. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc. Power consumption in packet radio networks. *Theoretical Computer Science*, 243(1-2):289–305, 2000.
40. S. Kumar, T.-H. Lai, and J. Balogh. On k-coverage in a mostly sleeping sensor network. *Wireless Networks*, 14:277–294, 2008.
41. W. Kieß and M. Mauve. A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Networks*, 5:324–339, 2007.
42. X.-Y. Li. Topology control in wireless ad hoc networks. In S. Basagni, M. Conti, S. Giordano, and I. Stojmenović, editors, *Mobile Ad Hoc Networking*, chapter 6, pages 175–204. IEEE Press, New York, NY, 2004.
43. J.-Y. LeBoudec and M. Vojnović. The random trip model: Stability, stationary regime, and perfect simulation. *IEEE/ACM Transactions on Networking*, 14:1153–1166, 2006.
44. S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proceeding of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1380–1387, 2001.
45. M. D. Penrose. The longest edge of the random minimal spanning tree. *The Annals of Applied Probability*, 7:340–361, 1997.
46. M. D. Penrose. On the k-connectivity for a geometric random graph. *Random Structures & Algorithms*, 15(2):145–164, 1999.
47. M. D. Penrose. *Random Geometric Graphs*. Oxford University Press, Oxford, 2003.
48. J. Pitman. *Probability*. Springer, New York, NY, 1999.
49. R. Rajaraman. Topology control and routing in ad hoc networks: A survey. *SIGACT News*, 33:60–73, 2002.
50. E. M. Royer and C-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications*, 7:46–55, 1999.
51. P. Santi. The critical transmitting range for connectivity in mobile ad hoc networks. *IEEE Transactions Mobile Computing*, 4(3):310–317, 2005.
52. P. Santi. Topology control in wireless ad hoc and sensor networks. *ACM Computing Surveys*, 37:164–194, 2005.
53. P. Santi and D. M. Blough. The critical transmitting range for connectivity in sparse wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):25–39, 2003.
54. P. Santi, D. M. Blough, and F. S. Vainstein. A probabilistic analysis for the range assignment problem in ad hoc networks. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing, (MobiHoc)*, pages 212–220, 2001.
55. A. Sen and M. L. Huson. A new model for scheduling packet radio networks. In *Proceedings of the 15th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, IEEE Computer Society, Los Alamitos, CA, 1997.
56. S. Schmid and R. Wattenhofer. Algorithmic models for sensor networks. In *Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS)*, IEEE Computer Society, Los Alamitos, CA, 2006.
57. S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman. A taxonomy of sensor network communication models. *Mobile Computing and Communication Review*, 6:28–36, 2002.
58. J. Tian, J. Hahner, C. Becker, I. Stepanov, and K. Rothermel. Graph-based mobility model for mobile ad hoc network simulation. In *Proceedings. 35th Annual Simulation Symposium*, IEEE Computer Society, Los Alamitos, CA, 2002.
59. E. J. van Leeuwen. *Optimization and Approximation on Systems of Geometric Objects*. PhD thesis, Universiteit van Amsterdam, 2009.
60. W. Wang, V. Srinivasan, and K.-C. Chua. Trade-offs between mobility and density for coverage in wireless sensor networks. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking (MobiCom)*, pages 39–50, ACM, New York, 2007.

61. P.-J. Wan and C.-W. Yi. Asymptotic critical transmission radius and critical neighbor number for -connectivity in wireless ad hoc networks. In J. Murai, C. E. Perkins, and L. Tassiulas, editors, *Proceedings of the 5th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing*, pages 1–8. ACM, New York, NY, 2004.
62. P.-J. Wan and C.-W. Yi. Coverage by randomly deployed wireless sensor networks. *IEEE Transaction on Information Theory*, 52(6):2658–2669, 2006.
63. X.Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *Proceedings of ACM SenSys*, pages 28–39, 2003.
64. J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *Proceeding of he 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, IEEE Computer Society, Los Alamitos, CA, 2003.
65. J. Yoon, M. Liu, and B. Noble. Sound mobility models. In D. B. Johnson, A. D. Joseph, and N. H. Vaidya, editors, *Proceedings of the Ninth Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 205–216. ACM, New York, NY, 2003.
66. J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52:2292–2330, 2008.
67. F. Zhao and L. Guibas, editors. *Distributed Environmental Monitoring Using Random Sensor Networks*, vol. 2634, *Lecture Notes in Computer Science*. Springer, 2003.
68. H. Zang and J. Hou. On deriving the upper bound of alpha-lifetime for large sensor networks. In J. Murai, C. E. Perkins, and L. Tassiulas, editors, *Proceedings of the 5th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing*, pages 16–24. ACM, New York, NY, 2004.
69. Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges. *IEEE Communications Surveys and Tutorials*, 8(1-4):24–37, 2006.

Chapter 7
Networked Distributed Source Coding

Shizheng Li and Aditya Ramamoorthy

Abstract The data sensed by different sensors in a sensor network is typically correlated. A natural question is whether the data correlation can be exploited in innovative ways along with network information transfer techniques to design efficient and distributed schemes for the operation of such networks. This necessarily involves a coupling between the issues of compression and networked data transmission that have usually been considered separately. In this work we review the basics of classical distributed source coding and discuss some practical code design techniques for it. We argue that the network introduces several new dimensions to the problem of distributed source coding. The compression rates and the network information flow constrain each other in intricate ways. In particular, we show that network coding is often required for optimally combining distributed source coding and network information transfer and discuss the associated issues in detail. We also examine the problem of resource allocation in the context of distributed source coding over networks.

7.1 Introduction

There are various instances of problems where correlated sources need to be transmitted over networks, e.g., a large-scale sensor network deployed for temperature or humidity monitoring over a large field or for habitat monitoring in a jungle. This is an example of a network information transfer problem with correlated sources. A natural question is whether the data correlation can be exploited in innovative ways along with network information transfer techniques to design efficient and distributed schemes for the operation of such networks. This necessarily involves a coupling between the issues of compression and networked data transmission that have usually been considered separately (see Fig. 7.1 for an illustration).

S. Li (✉)
Iowa State University, Ames, IA, USA
e-mail: szli@iastate.edu

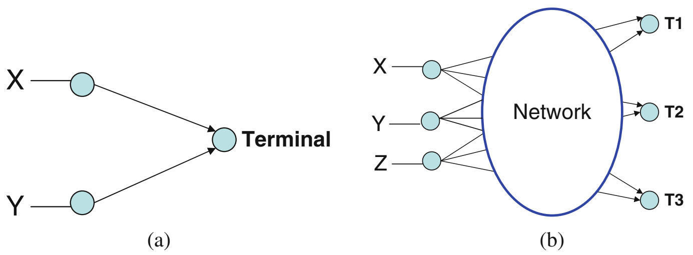


Fig. 7.1 (**a**) Classical Slepian–Wolf problem with sources X and Y with direct links to a terminal. (**b**) Practical scenario with multiple sources and terminals communicating over a network with link capacity and cost constraints. The joint problem of distributed source coding and network information transfer introduces various issues that are overviewed in this work

The correlation in a sensor network can be exploited in multiple ways. One can consider protocols where sensor nodes exchange information among themselves, compress the information, and then transmit the compressed bits to the terminal. At the other extreme, the sensors may operate independently. Intuitively, one would expect that the first scenario would be significantly better from a compression perspective. A surprising and groundbreaking result of Slepian and Wolf [1] shows that in fact under certain situations, the case in which the sensors act independently can be as efficient as the case in which the sensors do communicate with each other. The work of [1] introduced the idea of distributed source coding and demonstrated the existence of encoders and decoders that could leverage the correlation without needing explicit cooperation between the sources.

In this chapter we review various ideas in distributed source coding that are interesting within the context of sensor networks. We begin by an overview of the basic concepts and an outline of certain practical code constructions that have been the focus of much work recently. Next, we examine distributed source coding in a network context. The network introduces several dimensions to the problem of distributed source coding that do not exist in the classical case. It may be tempting to argue that one could simply find paths in the network that act as the direct links in the classical problem, assuming that the paths have enough capacity. However, such an approach is not optimal. The compression rates and the network information flow constrain each other in intricate ways. In particular, it turns out that network coding [2] is essential in certain cases for optimality. Interestingly enough, the flavor of results in this area depends upon the number of sources and terminals in the network. We survey these in a fair amount of detail in this chapter and examine the relationship between network coding and distributed source coding.

The issue of resource allocation is very important in the field of networking. For example, optimal routing of packets that maximizes some utility function of the network is a well-investigated issue in the field of networking [3]. Several techniques for solving these problems in a distributed manner have been studied in the literature [4]. In this chapter we discuss resource allocation problems in the context of transmitting correlated sources over a network. The main difference here is that one needs to jointly allocate the rates and the flows in the network. In particular, the network capacity region and the feasible rate regions interact in non-trivial ways.

This chapter is organized as follows. We discuss the basics of distributed source coding in Sect. 7.2 and introduce the problem of networked distributed coding in Sect. 7.3. Sect. 7.4 presents the discussion for the case of networks with a single terminal and Sect. 7.5 considers the case of networks with multiple terminals.

7.2 Basics of Distributed Source Coding

A sensor network consists of various sensors that monitor some physical phenomenon, e.g., an agricultural sensor network may be deployed in a field for temperature or humidity monitoring. In this chapter we will use the terms sensor and source interchangeably. Furthermore, a sensor output at a given time is assumed to be a random variable. Hence, over time, the observations of a sensor can be treated as a vector of random variables. We assume that the source outputs a sequence of independent and identically distributed (i.i.d.) random variables. While this assumption may not hold in a strict sense, we will see that it serves to simplify our exposition. Many of the results discussed in this chapter also hold for the case of sources with memory. However, we will not discuss them here.

Formally, we denote n successive realizations of a source X by $X_1, X_2, \ldots, X_n$, such that their joint distribution $p(X_1, \ldots, X_n) = \Pi_{i=1}^{n} p(X_i)$. If there is another correlated source Y, the joint distribution $p(X_1, Y_1, X_2, Y_2, \ldots, X_n, Y_n) = \Pi_{i=1}^{n} p(X_i, Y_i)$, i.e., at a given time instant, the sources are correlated but across time they are independent.

In a sensor network, the main problem is to convey either the sensor readings or their functions (e.g., mean, variance) to a terminal. The transmission protocol needs to be efficient in terms of the number of bits transmitted. If the correlation between the sources is ignored and if the terminal needs to recover the source without any distortion, the compression rate should be at least the entropy [5, 6] of the source. For example, if there are two sources X and Y, this implies that the terminal needs to receive $H(X) + H(Y)$ bits per unit time for recovering both X and Y.

Clearly, if there is correlation across sensors, the overall bit rate required for transmission to the terminal can be reduced. This is certainly feasible if the sources communicate with each other. The famous result of Slepian and Wolf [1] shows that distributed source coding, where the sources do not communicate with each other, can be as efficient as the case in which the sources communicate with each other.

7.2.1 Slepian–Wolf Theorem

Consider two sources X and Y. Let R_X and R_Y denote the rates at which the sources operate. This means that the sources X and Y transmit R_X and R_Y bits per unit time to the terminal.

Theorem 1 (Slepian–Wolf Theorem [1]) *Consider memoryless correlated sources X and Y from finite-sized alphabets $\mathcal{X}, \mathcal{Y}$ respectively, with joint distribution $p(X, Y)$. Suppose that*

$$\begin{aligned} R_X &\geq H(X|Y) \\ R_Y &\geq H(Y|X) \\ R_X + R_Y &\geq H(X, Y) \end{aligned}$$

There exist encoding functions $f_1 : \mathcal{X}^n \to \{1, 2, \ldots, 2^{nR_X}\}$ at source X and $f_2 : \mathcal{Y}^n \to \{1, 2, \ldots, 2^{nR_Y}\}$ at the source Y and a decoding function $g : \{1, 2, \ldots, 2^{nR_X}\} \times \{1, 2, \ldots, 2^{nR_Y}\} \to \mathcal{X} \times \mathcal{Y}$ at the terminal, such that the terminal is able to recover the source sequences with vanishing error probability as n goes to infinity. Conversely, if R_X, R_Y do not satisfy those conditions, it is impossible to recover the sources with vanishing error probability.

The implication of the Slepian–Wolf theorem is rather surprising and profound. Intuitively, it is clear that there is no hope of compressing the sources to a rate of less than $H(X, Y)$ even if they communicate with each other. The Slepian–Wolf theorem shows that in fact one can do this even when the sources do not communicate with each other.

The achievability proof goes as follows. A length n X-sequence is compressed to a binary vector of length nR_X by encoding function f_1 that is chosen at random. This process is referred to as random binning [6] in the literature, as each sequence is assigned a bin whose index is determined by f_1. Similarly, f_2 returns the bin index of a Y-sequence. At the terminal, suppose bin indices (i, j) are received. The decoding function finds all the length n sequences $\mathbf{x}, \mathbf{y}$ such that $f_1(\mathbf{x}) = i$, $f_2(\mathbf{y}) = j$ and find the pair of sequences that are most likely to have been transmitted. When n is large, with high probability, such sequence pair is the actual transmitted sequence pair. In other words, the error probability is vanishing as n goes to infinity.

The rates satisfying conditions are called achievable rates and they form a region in the 2-D plane shown in Fig. 7.2.

The two corner points on the boundary are interesting. They correspond to a rate allocation $(R_X, R_Y) = (H(X), H(Y|X))$ or $(R_X, R_Y) = (H(X|Y), H(Y))$.

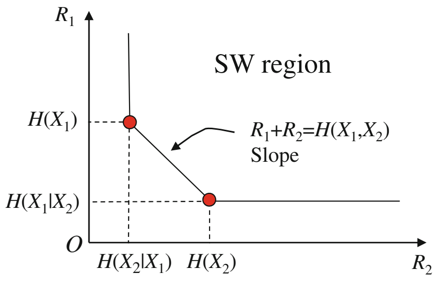


Fig. 7.2 Slepian–Wolf Region in the case of two sources X and Y

In order to achieve one of these points, say the first one, since $R_X = H(X)$, any lossless compression scheme can be used to compress **x**. Then, **x** is used as *side information* to help decode **y** at the decoder. The rate of Y is $H(Y|X)$, i.e., the amount of uncertainty given X.

Code design in the case when side information is available at the decoder is called the *asymmetric* Slepian–Wolf coding problem [7]. Code design for achieving any general (non-corner) point is called the *symmetric* Slepian–Wolf coding problem. There are many practical code designs for both asymmetric coding and symmetric coding when we have only two sources. In general, asymmetric Slepian–Wolf coding is easier than the symmetric case, because of a certain equivalence with channel coding, that we will discuss shortly. We refer the reader to [7] and the references therein for detailed descriptions.

The theorem above is stated for two sources. In general, when there are N sources, we have a generalized Slepian–Wolf theorem [8]. Suppose the sources $X_1, X_2, \ldots, X_N$ are generating i.i.d. symbols according to the joint probability distribution $p(X_1, X_2, \ldots, X_N)$. Let R_i denote the rate for source X_i and S denote a nonempty subset of node indices: $S \subseteq \{1, 2, \ldots, N\}$. Let X_S denote the set of random variables $\{X_i : i \in S\}$. If the rate vector $(R_1, R_2, \ldots, R_N)$ satisfies

$$\sum_{i \in S} R_i \geq H(X_S | X_{S^c}) \text{ for all } S \neq \emptyset$$

the decoder is able to recover all sources error free (asymptotically). Conversely, if the rates do not satisfy the condition, lossless recovery is impossible. When there are multiple sources, practical code design is a challenging problem. Some coding schemes exist, e.g., [9–11], but they either suffer suboptimal rate or have strong assumptions on the correlation model.

7.2.2 *Equivalence Between Slepian–Wolf Coding and Channel Coding*

The proof of the Slepian–Wolf theorem is information theoretic in nature and the corresponding achievability scheme requires exponential (in n) complexity decoding in general. For the case of two sources, and asymmetric Slepian–Wolf coding, Wyner [12] discovered the relation between channel coding and Slepian–Wolf coding. Most existing work on Slepian–Wolf coding for two sources relies on Wyner's idea and exploits powerful channel codes such as Turbo codes and LDPC codes [13–19]. Here, we introduce the basic ideas for asymmetric Slepian–Wolf coding.

First we review the concepts of channel coding [20], especially on linear block codes. A (n, k) linear block code over a finite field $GF(q)$ maps each message of length k (i.e., a k-length vector $\in GF(q)$) to a codeword **c** of length n (i.e., an n-length vector $\in GF(q)$). The codeword is transmitted through a channel, which introduces an error **e**. The receive vector is $\mathbf{r} = \mathbf{c} + \mathbf{e}$ (addition over $GF(q)$), where

$\mathbf{e}$ denotes the error vector. The decoder takes $\mathbf{r}$ as input and attempts to find the correct $\mathbf{c}$. In classical coding theory, the errors are modeled according to their Hamming weight, i.e., the number of nonzero elements in $\mathbf{e}$. An important design parameter of a code is the minimum Hamming distance d (the number of positions where two codewords take different values). A code with minimum distance d is able to correct up to $\lfloor (d-1)/2 \rfloor$ errors, i.e., as long as the Hamming weight of $\mathbf{e}$, $wt(\mathbf{e}) \leq \lfloor (d-1)/2 \rfloor$, the decoder can find the error pattern $\mathbf{e}$ and the transmitted codeword $\mathbf{c}$.

The parity check matrix of a linear block code is a $(n-k) \times n$ matrix H such that $\mathbf{c}H^T = 0$ (matrix multiplication over $GF(q)$) for every codeword $\mathbf{c}$. A practical decoding algorithm for a linear block is called syndrome decoding. The decoder computes the syndrome of length $(n-k)$ $\mathbf{s} = \mathbf{r}H^T$. Since $\mathbf{r}H^T = \mathbf{c}H^T + \mathbf{e}H^T$, $\mathbf{s} = \mathbf{e}H^T$, implying that the syndrome only depends on the error pattern. It then attempts to find the $\mathbf{e}$ with the least weight. This can be done efficiently for specific classes of codes. For example, Berlekamp–Massey algorithm for BCH codes and Reed–Solomon codes [20] can be used to find the error pattern $\mathbf{e}$ from $\mathbf{s}$ as long as $wt(\mathbf{e}) \leq (d-1)/2$. Likewise, binary LDPC codes admit efficient decoding.

We now demonstrate that syndrome decoding can be applied to the asymmetric Slepian–Wolf coding problem. Assume that the source sequences $\mathbf{x}, \mathbf{y}$ have length n and the correlation model is that the Hamming distance between them is no more than t, i.e., they differ at most t positions. Suppose $\mathbf{y}$ is available at the decoder. At source X, we transmit $\mathbf{x}H^T$ to the terminal. The terminal computes $\mathbf{y}H^T + \mathbf{x}H^T = (\mathbf{x}+\mathbf{y})H^T = \mathbf{e}H^T$, where $\mathbf{e} = \mathbf{x}+\mathbf{y}$ is the difference between $\mathbf{x}$ and $\mathbf{y}$.[1] We know that $\mathbf{x}$ and $\mathbf{y}$ differ in at most t positions, so $wt(\mathbf{e}) \leq t$. The decoder is able to find $\mathbf{e}$ as long as the minimum distance of the channel code is at least $2t+1$ based on the discussions above. Once $\mathbf{e}$ is obtained, $\mathbf{x} = \mathbf{y} + \mathbf{e}$ can be easily computed. Thus, a length n vector $\mathbf{x}$ is compressed to a length $(n-k)$ vector $\mathbf{x}H^T$. Since the minimum distance of a code should satisfy Singleton bound $d \leq n-k+1$ [20], the length $n-k$ should be at least $2t$.

In order to establish a concrete relationship with Slepian–Wolf theorem, next we consider a probabilistic correlation model. Consider binary sources X and Y that are uniformly distributed. The correlation between them is that the probability that they are different is $p < 0.5$. In other words, each bit in the vector $\mathbf{e} = \mathbf{x} + \mathbf{y}$ is 1 with probability p and 0 with probability $1-p$. Then, $H(X|Y) = H_b(p)$,[2] and $H(X,Y) = 1 + H_b(p)$.

Now, consider the channel coding problem for the binary symmetric channel (BSC) with crossover probability p. The codeword $\mathbf{c}$ is transmitted and $\mathbf{r} = \mathbf{c} + \mathbf{e}$ is received and $\mathbf{e}$ is i.i.d. taking value 1 with probability p. The capacity of this channel is $1 - H_b(p)$ [6]. The receiver computes the syndrome $\mathbf{s} = \mathbf{r}H^T = \mathbf{e}H^T$. It can be shown that there exists an H and the decoding function $f_{\text{dec}}(\cdot)$ such that the code

[1] In this chapter, assume that the size of the finite field is a power of 2 so addition and subtraction are the same.

[2] $H_b(p)$ is the binary entropy function defined as $H_b(p) = -p\log_2 p - (1-p)\log_2(1-p)$.

rate $k/n \to 1 - H_b(p)$ as $n \to \infty$ and the decoding error can be made arbitrarily small [21]. Such a code is called a capacity-achieving code.

In an asymmetric Slepian–Wolf coding setting, suppose that the decoder knows $\mathbf{y}$. Let $R_Y = H(Y) = 1$ and apply any lossless entropy coding scheme [6], $\mathbf{y}$ can be recovered at the terminal. Take the parity check matrix of a capacity-achieving code H and the source X transmits $\mathbf{x}H^T$. The terminal finds the estimate of $\mathbf{x}$,

$$\hat{\mathbf{x}} = \mathbf{y} + f_{\text{dec}}(\mathbf{x}H^T + \mathbf{y}H^T)$$

The probability that $\hat{\mathbf{x}} \neq \mathbf{x}$ is arbitrary small. Note that the length of vector transmitted by source X is $n - k$, so the rate

$$R_X = (n-k)/n = 1 - k/n = H_b(p) = H(X|Y)$$

Thus, using a capacity-achieving channel code, we can achieve the corner point $(H(X|Y), H(Y))$ of the Slepian–Wolf region.

In practice, LDPC codes [22] come very close to the BSC capacity. The belief propagation algorithm (BPA) [22] acts as the decoding function $f_{\text{dec}}(\cdot)$. Note that in the channel coding setting, the belief propagation algorithm is designed to output a codeword $\mathbf{c}$ with 0 syndrome, whereas in the distributed source coding setting, the BPA needs to be modified so that it outputs a vector satisfying a given syndrome. More generally, even if the correlation model cannot be viewed as a binary symmetric channel, we can provide proper initialization to the BP algorithm according to the correlation model. Turbo codes can also be used to achieve compression via puncturing at the encoder; the extrinsic information exchange at the decoder exploits the correlation between the sources [23–25].

The equivalence in the asymmetric case does not carry over in a straightforward manner to the symmetric case. However, an approach called source splitting [26, 27] allows us to transform the symmetric Slepian–Wolf coding problem for N sources to an asymmetric (corner point) problem where there are $2N - 1$ sources.

7.2.3 Distributed Source Coding with a Fidelity Criterion

In the previous sections we considered the problem of lossless reconstruction. In many practical applications, we may allow a certain amount of distortion in the recovery process. In lossy multiterminal source coding, each source encodes its own data at a certain rate and transmits it to the terminal. The terminal tries to recover all the sources under a fidelity criterion. The fidelity is measured with respect to a distortion metric.

More specifically, the encoders observe source sequences $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ emitted by the sources $X_1, X_2, \ldots, X_N$ and encode them at rate $R_1, R_2, \ldots, R_N$ separately (with no communication between the encoders). Given distortion metrics, $\mathbf{D} = (D_1, D_2, \ldots, D_N)$ for each source, we hope to find the region $\mathcal{R}(\mathbf{D})$ of all rates $\mathbf{R} = (R_1, R_2, \ldots, R_N)$ that allow the decoder to reconstruct $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_N$ such that the expected distortion between $\mathbf{x}_i$ and $\hat{\mathbf{x}}_i$ is less than D_i for all $i = 1, 2, \ldots, N$. However, the general region even in the case of very specific distortion metrics remains unknown.

The inner bound for a given problem refers to a set of rates that can be shown to be achievable. The outer bound refers to a set of rates that are not achievable under any strategy. Some inner/outer bounds for the general problem can be found in [28–30]. In most cases the inner and outer bounds do not meet, i.e., the exact region is unknown. A tighter outer bound was obtained recently [31] and some insights on the optimal encoders and decoders are given in [32]. The quadratic Gaussian case was considered in [33, 34], where the rate-distortion regions for several special cases are determined. Practical code design for multiterminal rate-distortion problems is discussed in [35, 36].

Next we discuss two special cases of multiterminal source coding problems, for which the rate-distortion regions are relatively well studied.

7.2.3.1 Wyner–Ziv Coding

Consider two correlated sources X and Y that follow joint distribution $p(X, Y)$. The source sequence $\mathbf{x}$ needs to be encoded without knowing $\mathbf{y}$ and transmitted to the decoder, at which side information $\mathbf{y}$ is available. Let the distortion between two n length sequences $\mathbf{x}$ and $\hat{\mathbf{x}}$ be measured as $\frac{1}{n}\sum_{i=1}^{n} d(x_i, \hat{x}_i)$, where d is a non-negative function. The rate-distortion function $R_{WZ}(D)$ gives the minimum required rate such that the expected distortion between the actual source sequence $\mathbf{x}$ and the decoder output $\hat{\mathbf{x}}$ is upper bounded by D, i.e., $E\left(\frac{1}{n}\sum_{i=1}^{n} d(x_i, \hat{x}_i)\right) \leq D$. Clearly, if $D = 0$, it is the special instance of Slepian–Wolf problem at corner point $(H(X|Y), H(Y))$. In general, the rate-distortion function was shown by Wyner and Ziv [37] to be

$$R_{\mathrm{WZ}}(D) = \min_{P_{U|X}(\cdot), f(\cdot): E(d(X, f(U,Y))) \leq D} I(X; U) - I(Y; U)$$

where U is an auxiliary random variable and is such that $U \rightarrow X \rightarrow Y$, i.e., U, X, Y form a Markov chain and the expectation is taken over the joint distribution of X, Y, U. The function f is the decoding function.

In the Slepian–Wolf setting (i.e., $D = 0$), we know that minimum required rate is $H(X|Y)$, whether or not Y is available at the X encoder. When $D > 0$, let us denote the rate required when Y is available at the source encoder as $R_{X|Y}(D)$. It can be shown that in some cases $R_{X|Y}(D) < R_{WZ}(D)$. In other words, we may lose efficiency when encoding correlated sources separately rather than jointly when $D > 0$. In the special case when the sources are correlated by $X = Y + Z$ where Y and Z are both Gaussian and Z is independent of Y, $R_{X|Y}(D) = R_{WZ}(D)$ [37]. In many other correlation models, the equality does not hold.

Practical coding schemes for the Wyner–Ziv problem based on nested codes [38, 39] are known. Nested lattice codes can be used in the quadratic Gaussian case and can be shown to achieve the Wyner–Ziv bound. Other practical Wyner–Ziv code designs include trellis-based codes [13], nested coding followed by Slepian–Wolf coding [40], quantization followed by Slepian–Wolf coding [41, 42]. The discussion of these techniques is beyond the scope of this survey.

7.2.3.2 The CEO Problem

In the CEO problem [43], there is one source X and N encoders that do not observe the source directly. Instead, each encoder observes a corrupted version of X, denoted as Y_i, $i = 1, 2, \ldots, N$. The Y_i's are assumed to be conditionally independent given X. The encoder encodes $\mathbf{y}_i$ at rate R_i and such that the total encoding rate is $\sum_{i=1}^{N} R_i \leq R$. The decoder finds the $\hat{\mathbf{x}}$ (the estimate of $\mathbf{x}$,), based on the encoded codewords. The aim is to find the rate-distortion function $R(D)$, i.e., the minimum total encoding rate needed such that the expected distortion between $\mathbf{x}$ and $\hat{\mathbf{x}}$ is at most D. This is analogous to a situation when a Chief Executive (or Estimation) Officer obtains information from N agents and wants to estimate the source sequence $\mathbf{x}$ that he or she is interested in. In a sensor network application, we can think of the data fusion center acting as the CEO and the sensors act as the agents. The problem formulation takes into account the noise in the sensing procedure. The original paper [43] determined the asymptotic behavior of the error frequency when $R \to \infty$ for discrete memoryless source. The quadratic Gaussian case of the CEO problem, where X is Gaussian and the observation noises $Y_i - X$ are independently Gaussian distributed, is studied in [44–46] and the rate-distortion function is determined in [45, 46].

7.3 Networked Distributed Source Coding: An Introduction

In the previous sections we have discussed the classical Slepian–Wolf result and its lossy variants. Note that so far we have assumed that there is a direct noiseless link between the sources and the terminal. This is a useful simple case to analyze and captures the core of the problem as far as the basic concept of distributed source coding is concerned. However, in a practical sensor network we expect that the sensors will be communicating with the terminal over a network, possibly with the help of various relay nodes. Therefore, it is natural to investigate whether the process of information transmission over the network influences the compression and vice versa. Our network model represents a wireline network or a wireless network with medium access control (MAC) protocols that make the channels look independent (we discuss the network model in more detail later). In this part of the chapter, we overview relatively recent work that has contributed toward our understanding of this field.

The problem of networked distributed source coding differs from the classical problem in the following ways.

- *Suboptimality of separation between distributed source coding and network information transfer.*
  Note that the problem of distributed source coding over networks would be a straightforward extension of the classical Slepian–Wolf problem if one could essentially "simulate" the presence of direct links between the sources and the terminal. Indeed, one could encode the sources using a classical Slepian–Wolf code and simply "flow" the encoded bits over the appropriate paths. This would amount to separating the tasks of distributed source code design and the problem of network information transfer. It turns out that such a strategy is suboptimal in general.
- *Issues of optimal resource allocation over the network.*
  The network introduces several issues with respect to the allocation of rates and flows such that they are in some sense "optimal" for the operation of a network. For example, in sensor networks, the problem of deciding the appropriate paths over which the data needs to flow for minimum energy or maximum lifetime [47] is of interest. In the context of correlated sources, these issues become more complicated since one needs to jointly optimize the rates and the flows.

Our model of a network is a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. There is a set of source nodes $S \subset V$ that observes the sources and a set of terminals $T \subset V$ that needs to reconstruct the sources. An edge (v_1, v_2) is a communication channel which allows information transmission from v_1 to v_2. The channel can be noisy or deterministic (but typically capacity constrained). The different channels in the network are in general dependent, e.g., in a wireless network, broadcast, and interference induces dependence between different channels. However, characterizing the capacity region in such scenarios, even with independent messages, has proved to be a difficult task [6]. In fact, in many practical situations, protocols such as time division multiple access-TDMA, frequency division multiple access-FDMA are used to provide the semblance of independent channels. In a wireline network, the channels are typically independent. In the discussion in the sequel, we will mostly work under the assumption that the channels are independent. It turns out that the results in this area depend critically on the number of terminals in network. Accordingly, we divide the discussion into two different sections. In Sect. 7.4 we review the results for the single terminal case and in Sect. 7.5 we review the corresponding results for multiple terminals.

7.4 Networked Distributed Source Coding: Single Terminal

In networks with a single terminal, under the assumption that the channels are independent, Han [48] gave necessary and sufficient conditions for a network to be able to transmit the correlated sources to the sink. A simple achievable transmission scheme was proposed and its optimality was proved. Barros et al.[49] obtained the same result under a more general encoding model, where the form of joint source/channel coding can be arbitrary and the coding can be across multiple blocks. The achievability proof is almost the same as [48] and the converse is proved in a different manner and is stronger because of the more general coding model.

Suppose that there are $N + 1$ nodes $v_0, v_1, \ldots, v_N$ in the network observing sources $X_0, X_1, \ldots, X_N$. The graph $G(V, E)$ is complete and each edge (v_i, v_j) is a discrete memoryless channel with capacity C_{ij}. Note that the source entropy could be zero and the capacity could also be zero, so realistic networks can easily fit into this general framework. Node v_0 is the sink that wants to reconstruct the sources $X_1, \ldots, X_N$.

The proposed transmission scheme is very simple and intuitive. Apply good channel codes to each channel so that we can model every edge (v_i, v_j) as a noiseless link with capacity C_{ij}. Each node performs Slepian–Wolf coding at rate R_i. Next, the Slepian–Wolf coded bits need to be routed to the sink v_0. Knowing the rates at each source node, we can find a feasible flow that supports rate R_i at source node v_i and terminates at sink node v_0 as follows.

Add a virtual supersource s^* and introduce an edge (s^*, v_i) with capacity $C_{s^*i} = R_i$ for $i = 1, \ldots, N$. Then compute the max-flow between s^* and v_0 [50]. This returns a flow assignment on each edge. The Slepian–Wolf coded bits are routed according to the flow assignment to v_0.

The node v_0 receives all Slepian–Wolf coded bits and jointly decodes all the sources $X_1, X_2, \ldots, X_N$. In order to reconstruct the sources, the rate vector $(R_1, \ldots, R_N)$ needs to be in the Slepian–Wolf region, i.e., for any nonempty subset of $\{0, \ldots, N\}$, S, such that $0 \in S^c$ (since X_0 is available at v_0 as side information and is not encoded)

$$\sum_{i \in S} R_i \geq H(X_S | X_{S^c}) \tag{7.1}$$

In order to successfully find the flow of value $\sum_{i=1}^{N} R_i$ from s^* to v_0, we need the capacity of any cut separating s^* and v_0 to be greater than $\sum_{i=1}^{N} R_i$. Note that a cut separates the source nodes into S and S^c, where $S \subseteq \{0, \ldots, N\}, 0 \in S^c$ but s^* does not connect to v_0, its capacity is $\sum_{j \in S^c \setminus \{0\}} C_{s^*j} + \sum_{i \in S, j \in S^c} C_{ij} = \sum_{j \in S^c \setminus \{0\}} R_j + \sum_{i \in S, j \in S^c} C_{ij}$. Thus, as long as

$$\sum_{i \in S} R_i \leq \sum_{i \in S, j \in S^c} C_{ij} \tag{7.2}$$

for all nonempty subset S of $\{0, \ldots, N\}$ such that $0 \in S^c$, the flow exists. This is illustrated in Fig. 7.3. Moreover, if

$$H(X_S | X_{S^c}) \leq \sum_{i \in S, j \in S^c} C_{ij} \tag{7.3}$$

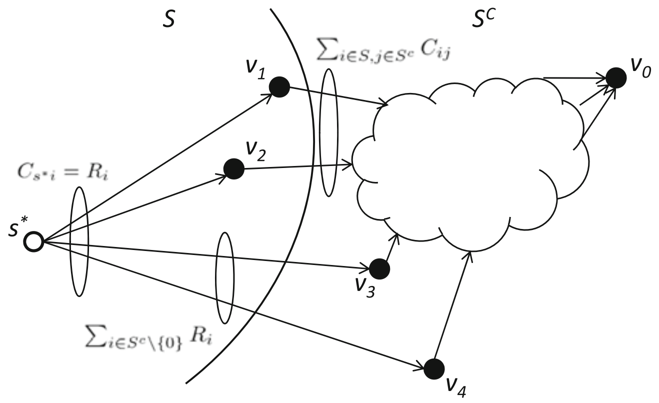

S
S^c
$\sum_{i\in S, j\in S^c} C_{ij}$
v_1
v_0
v_2
$C_{s^*i} = R_i$
s^*
v_3
$\sum_{i\in S^c\setminus\{0\}} R_i$
v_4

Fig. 7.3 Illustration of the sufficient condition for routing Slepian–Wolf coded bits to the terminal. s^* is the supersource. The cut of interest contains v_1, v_2 in S and v_3, v_4 in S^c. The cut capacity is $\sum_{j\in S^c\setminus\{0\}} R_j + \sum_{i\in S, j\in S^c} C_{ij}$, and it should be no less than $\sum_{i=1}^{N} R_i$. Thus, $\sum_{i\in S, j\in S^c} C_{ij} \geq \sum_{i\in S} R_i$

there exists a rate allocation satisfying (7.1) and (7.2) [48]. Therefore, (7.3) is a sufficient condition for the single sink data collection with Slepian–Wolf coding.

Conversely, it is proved that the above condition is the necessary condition for successful transmission under any joint coding scheme, i.e., if the capacity does not satisfy this condition, the sink cannot recover the sources losslessly, under any kind of coding scheme. Note that the proposed achievability scheme separates source coding, channel coding, and routing. The converse part implies that it is optimal to separately perform channel coding, distributed source coding, and treat the Slepian–Wolf coded bits as commodities and route to the terminal. The main theorem in [49] can also be viewed as a general source–channel separation theorem for networks with one terminal, with independent channels. It implies that the source coding, routing, and channel coding can be put into different layers of the protocol stack separately.

We emphasize, however, that such a separation does not hold in general, i.e., when there are more terminals. As we shall see in Sect. 7.5, even when the channels are independent, if we have more terminals, the compression rates and the network flows are closely coupled.

7.4.1 Optimal Rate and Flow Allocation

From the discussion in previous sections, it is clear that distributed source coding can compress the data effectively. In this section, we discuss resource allocation problems for networked distributed source coding.

A natural resource allocation problem is to determine the rate at which each source should be encoded, and the corresponding flows such that some network metric is optimized. For simplicity, we first consider the case when there are direct channels between the sources and the sink.

7.4.1.1 Direct Source–Sink Channels

Suppose the sources communicate to the sink directly. We consider two metrics as follows.

1. Sum rate minimization: In this case we consider noiseless source–sink channels and seek to find a feasible rate vector that minimizes $\sum_{i=1}^{N} R_i$.
2. Sum power minimization: Here we assume orthogonal additive white Gaussian noise (AWGN) channels between the sources and the sink and seek to minimize the total power min $\sum_{i=1}^{N} P_i$ (where P_i is the power of the ith source), expended in ensuring that the sources can be reconstructed at the terminal.

For the noisy channel case, the source nodes first use Slepian–Wolf codes to encode the sources. As long as each rate is less than the channel capacity the sources can be recovered losslessly at the terminal (assuming capacity-achieving codes are used). The capacity of the channel between node i and the sink with transmission power P_i and channel gain γ_i is $C_i(P_i) \equiv \log(1 + \gamma_i P_i)$, where the noise power is normalized to one and channel gains are constants that are known to the terminal. Thus, the rate R_i should satisfy $R_i \leq C_i(P_i)$. It is easy to see at the optimum, the sensor node should transmit at the capacity, i.e., $R_i^* = C_i\left(P_i^*\right)$. Thus, the power assignment is given by the inverse function of C_i which we denote by $Q_i(R_i)$, i.e., $P_i^* = Q_i\left(R_i^*\right) = \left(2^{R_i^*} - 1\right)/\gamma_i$. Once we know the optimal rate assignment R_i^* we know the power assignment P_i^* and vice versa. Therefore, the objective function of the sum power minimization problem can also be written as

$$\min \sum_{i=1}^{N} \left(2_i^R - 1\right)/\gamma_i$$

For both problems, if N-dimensional Slepian–Wolf codes are used, the rates should be in the N-dimensional Slepian–Wolf region, which is denoted by SW_N. Then, the sum rate minimization problem can be written as

$$\min_{R_1,\ldots,R_N} \sum_{i=1}^{N} R_i$$
$$\text{subject to } (R_1, \ldots, R_N) \in SW_N$$

The solution to this problem is trivial, i.e., any point at the boundary of the N-dimensional Slepian–Wolf region is the optimal solution. In the sum power minimization problem, besides Slepian–Wolf region constraint, we also add peak power constraints for the transmission power of each sensor node, taking into account the fact that every sensor has limited transmission power in a wireless sensor network. Then, the problem is a convex optimization problem:

$$\begin{aligned} \min_{R_1,\ldots,R_N} \sum_{i=1}^{N} P_i = \sum_{i=1}^{N} (2^{R_i} - 1)/\gamma_i \\ \text{subject to } (2^{R_i} - 1)/\gamma_i \leq P_{\max}, \forall i \\ (R_1, \ldots, R_N) \in SW_N \end{aligned}$$

This problem can be efficiently solved by, for example, interior-point methods [51].

In practice we do need to impose additional restrictions on the set of feasible rate vectors. This is primarily because the problem of practical code design for the N-dimensional Slepian–Wolf region remains open. It is fair to say that at present, we only know how to design Slepian–Wolf codes for two sources. Thus, it makes sense to impose "pairwise" constraints on the rate vectors, so that two sources can be decoded together. Given the state-of-the-art code designs for two sources case, we could perform encoding and decoding in a pairwise fashion. Before the transmission starts, we determine the source pairs that are jointly decoded together each time and determine the rates of the sources and the corresponding codes. During the transmission, the sources encode the message separately (without communication with other sources) using the preassigned code and the sink performs joint decoding for two nodes each time according to the preassigned combinations. We call this *pairwise distributed source coding*, which is simple and practical. The resource allocation problem is to determine the optimal pairing combinations and the rates for the sensors such that the sum rate or the sum power is minimized. This problem was first considered and solved using the notion of matching in undirected graph in [52]. Later, an improved solution using the notion of minimum weight arborescences and matching forests was proposed in [53] that we shall discuss below.

First, we consider the sum rate minimization problem. Note that any point on the slope of the Slepian–Wolf boundary achieves the minimum sum rate of two sources. Thus, for a pair of sources that will be decoded together, simply choosing the corner point as a rate allocation achieves minimum sum rate. Also note that a decoded source can be used as side information to help decode other sources at the terminal so that the rate of other sources being helped can be as low as the conditional entropy given the decoded source. Since we consider pairwise distributed source coding here and each time only two sources are involved in the decoding, we do not use more than one decoded sources as helper. We say a rate assignment has the pairwise property if it allows the terminal decode the sources in a pairwise fashion. Specifically, the rate assignment is said to satisfy the pairwise property if for each source $X_i, i = 1, 2, \ldots, N$, there exists an ordered sequence of sources $(X_{i_1}, X_{i_2}, \ldots, X_{i_k})$ such that

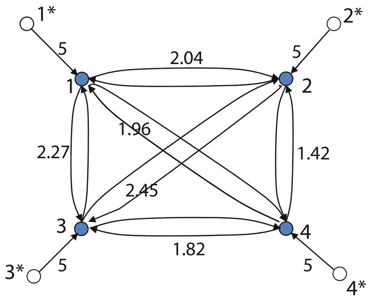
$$R_{i_1} \geq H(X_{i_1}) \tag{7.4}$$

$$R_{i_j} \geq H(X_{i_j}|X_{i_{j-1}}), \quad \text{for } 2 \leq j \leq k, \text{ and} \tag{7.5}$$

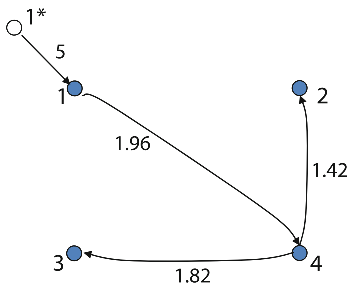
$$R_i \geq H(X_i|X_{i_k}) \tag{7.6}$$

Such a rate assignment allows the possibility that each source can be reconstructed at the decoder by solving a sequence of decoding operations at the SW corner points, e.g., for decoding source X_i one can use X_{i_1} (since $R_{i_1} \geq H(X_{i_1})$), then decode X_{i_2} using the knowledge of X_{i_1}. Continuing in this manner finally X_i can be decoded. We hope to find rate assignment with pairwise property and with minimum sum rate. Clearly, the optimal rate assignment satisfies conditions (7.4), (7.5), and (7.6) with equality. It is easy to see the sequential decoding procedure of a rate assignment with pairwise property that can be expressed on a tree. The nodes at the higher layer are decoded first and used as side information to help decode the nodes at the lower layer. If we assign edge weights to be entropies and conditional entropies, the weight of the tree is the sum rate. Therefore, this inspires us to find a tree with minimum weight on a proper defined graph.

Now we formally describe our approach. Construct a directed graph $G = (V, E)$ as follows. The node set V consists of N regular nodes: $1, 2, \ldots, N$ and N starred nodes $1^*, 2^*, \ldots, N^*$. The edge set E consists of edges $(i^* \to i)$ with weight $H(X_i)$ for all $i = 1, 2, \ldots, N$, and edges $(i \to j)$ with weight $H(X_j|X_i)$ for all $i, j = 1, 2, \ldots, N$. An example of G is shown in the left figure of Fig. 7.4. Define a subgraph G_{i^*} of G as a graph obtained from G by deleting all starred nodes except i^* and all edges of the form $(j^* \to j)$ for $j \neq i$. For each i, find a minimum weight directed spanning tree[3] on G_{i^*}. This tree gives a rate allocation: $R_i = H(X_i)$, $R_j = H(X_j|X_{\text{inc}(j)})$, where $inc(j)$ is the node such that edge $(inc(j) \to j)$ belongs to the tree. Each subgraph G_{i^*} gives a rate allocation by a minimum weight



(a)



(b)

Fig. 7.4 An example of the rate allocation algorithm. The *left* figure shows the graph G. The edge weights on the edges from node i^* to node i are individual entropies and the edge weights on the edges between regular nodes are conditional entropies. In this example, the individual entropies are the same. Thus, $H(X_i|X_j) = H(X_j|X_i)$ and we only label one number between regular nodes i and j. The *right* figure shows the minimum weight directed spanning tree found on G_{1^*} (**a**) The graph G; (**b**) The minimum weight directed spanning tree found on G_{1^*}

[3] A directed spanning tree (also called arborescence) of a directed graph $G = (V, A)$ rooted at vertex $r \in V$ is a subgraph T of G such that it is a spanning tree if the orientation of the edges is ignored and there is a path from r to all $v \in V$ when the direction of edges is taken into account. The minimum weight directed spanning tree can be found by a greedy algorithm in polynomial time [54].

directed spanning tree and the one with minimum weight gives the final optimal rate allocation of the network. Note that if each source has the same entropy, the weights of minimum weight directed spanning trees of G_{i^*} are the same for each i, so we only need to pick up an arbitrary subgraph G_{i^*} and find the assignment on it. Clearly, the resulting rate assignment has the pairwise property and is optimal. In the example in Fig. 7.4, each source has the same entropy and the minimum weight directed spanning tree rooted at node 1^* is shown in the right figure. The optimal rate allocation is $R_1 = H(X_1)$, $R_4 = H(X_4|X_1)$, $R_2 = H(X_2|X_4)$, and $R_3 = H(X_3|X_4)$. The corresponding decoding procedure is that first decode source X_1, and use X_1 as side information to help decode X_4. Then, X_4 is used as side information to help decode X_2 and X_3.

Next, we show some simulation results. Consider a wireless sensor network example in a square area where the coordinates of the sensors are randomly chosen and uniformly distributed in [0, 1]. The sources are assumed to be jointly Gaussian distributed such that each source has zero mean and unit variance (this model was also used in [55]). The parameter c indicates the spatial correlation in the data. A lower value of c indicates higher correlation. The individual entropy of each source is $H_1 = \frac{1}{2}\log(2\pi e\sigma^2) = 2.05$. In Fig. 7.5, we plot the normalized sum rate found by minimum weight spanning tree (MST) $R_{s0} \equiv \sum_{i=1}^{N} R_i/H_1$ vs. the number of sensors n. If no distributed source coding is used, i.e., the nodes transmit data individually to the sink, $R_i = H_1$ and $R_{s0} = N$. Clearly, by pairwise distributed source coding, the sum rate is reduced. We also plotted the optimal normalized sum rate when N-dimensional Slepian–Wolf code is used $H(X_1, \ldots, H_N)/H_1$ in the figure. It is interesting to note that even though we are doing pairwise distributed source coding, our sum rate is quite close to the theoretical limit which is achieved by N-dimensional distributed source coding.

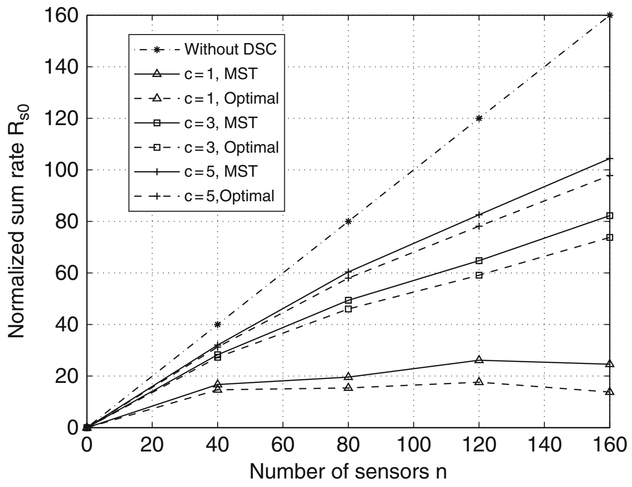


Fig. 7.5 Normalized sum rate vs. number of sensors

Now we consider the sum power minimization problem. Note that for a pair of sources that will be decoded together, the optimal rate allocation that minimizes the sum power of the pair is no longer a corner point but rather a particular point on the slope (which can be found by solving a simple optimization problem). For a node pair i and j, denote the optimal power allocation as $P^*_{ij}(i)$, $P^*_{ij}(j)$. We cannot simply choose the corner points and perform asymmetric Slepian–Wolf coding. We want some source pairs working at corner points while some others working at the optimal point on the slope of the 2-D SW region. Taking this into account, we generalize the concept of pairwise property. Recall that previously, under a rate assignment with pairwise property, the first source in a sequence is encoded at the rate of its entropy. Now we allow the first source in a decoding sequence to be paired with another node and encoded at the rate on the slope of the 2-D Slepian–Wolf region. The appropriate structure for finding the optimal resource allocation turns out to be one that combines the directed spanning tree and the matching. Such a structure is the *matching forest* first introduced in the work of Giles [56]. In fact, we are interested in a specific form of matching forest called *strict matching forest* (SMF). For the formal definitions, we refer the reader to [53]. Roughly speaking, a strict matching forest is a subgraph of a mixed graph[4] that connects every node only once. The SMF plays a role similar to the spanning tree in the sum rate minimization problem. The sequential decoding procedure of a rate assignment with generalized pairwise property can be expressed on a SMF. The node pairs connecting with undirected edges work at the slope of the Slepian–Wolf region and a symmetric coding scheme is used for them. The nodes that are connected with directed edge work at the corner point of the Slepian–Wolf region and the tail (origin) of a directed edge is used as side information to help decode the head (destination) of the edge. If we assign edge weights to be transmission powers, the weight of the SMF is the total transmission power.

Now we formally describe our approach. Construct a mixed graph $G = (V, E, A)$ as follows. The node set V consists of N regular nodes: $1, 2, \ldots, N$ and N starred nodes $1^*, 2^*, \ldots, N^*$. Recall that $Q_i(R)$ is the power consumed in transmission at rate R. For each $i = 1, 2, \ldots, N$, if $Q_i(H(X_i)) \leq P_{\max}$, add edge $(i^* \to i)$ with weight $Q_i(H(X_i))$. For each $i, j = 1, 2, \ldots, N$, if $Q_i(H(X_i|X_j)) \leq P_{\max}$, add edge $(j \to i)$ with weight $Q_i(H(X_i|X_j))$. For each pair i and j, if the optimal power allocation $P^*_{ij}(i)$, $P^*_{ij}(j)$ that minimizes the sum power of the pair of nodes exists, add undirected edge (i, j) with weight $P^*_{ij}(i) + P^*_{ij}(j)$. Then, find the minimum weight SMF on G, which gives the rate/power assignment with the generalized pairwise property and minimum sum power. It is shown in [53] that the problem of finding minimum weight SMF can be transformed and solved in polynomial time [57]. From the simulations we observe that in most cases, the optimal allocation is such that only one pair works on the slope and all other sources work at the corner points.

[4] "Mixed" graph refers to a graph with directed edges and undirected edges.

7.4.1.2 General Multihop Communication Between Sources and Sink

The resource allocation problem in a network with general topology and relay nodes was first considered by Han [48] and a similar formulation is given in [49]. We reformulate the problem as follows.

The network is given by a directed graph $G = (V, E, C)$, where $C = \{C_{ij} : (i, j) \in E\}$ is the capacity of each edge. Edge (i, j) is also associated with a weight w_{ij}. The cost of a flow of value z_{ij} going through the edge can be written as $F(z_{ij})w_{ij}$, where $F(\cdot)$ is a non-negative, increasing function. Then, the optimization problem can be written as

$$\begin{aligned}
&\min \sum_{(i,j)\in E} F(z_{ij})w_{ij} \\
\text{s.t.}\quad & 0 \le z_{ij} \le C_{ij}, \forall (i,j) \in E \quad \text{(capacity constraint)} \\
& \sum_{j|(i,j)\in E} z_{ij} - \sum_{j|(j,i)\in E} z_{ji} = \sigma_i, \forall i \in V \quad \text{(flow balance constraint)} \\
& (R_1, R_2, \ldots, R_N) \in SW_N \quad \text{(Slepian–Wolf constraint)}
\end{aligned}$$

where $\sigma_i = R_l$ if i is the lth source node, $\sigma_i = -\sum_{i=1}^{N} R_i$ if i is the sink and otherwise, $\sigma_i = 0$.

For simplicity, we can consider linear cost $F(z_{ij}) = z_{ij}$. Then, the above optimization is a linear program. If $F(\cdot)$ is a convex function, it is a convex optimization problem.

If there is no capacity constraint, the solution of the problem has a simple form and interpretation [58]. The basic idea is that in the absence of capacity constraints, there is no need to split the flow across different edges. Once a route (path) from a given source to the sink with minimum cost is found, the source simply routes all the data through that path. For example, suppose that the minimum cost path for source X_l is $\mathcal{P}^l$. Then for all edges (i, j) belonging to $\mathcal{P}^l$, we set $z_{ij} = R_l$. In this case, the cost of transmitting the data from X_l to the sink is $\sum_{e\in\mathcal{P}^l} F(R_l)w_e$. Thus, the overall cost function becomes

$$\min_{\{R_l,d_l\},l=1,2,\ldots,N} \sum_{l=1}^{N} F(R_l)d_l$$

where d_l is the total weight of path $\mathcal{P}^l$, i.e., $d_l = \sum_{e\in\mathcal{P}^l} w_e$. Solving this problem involves finding the optimal paths $\mathcal{P}^l, l = 1, 2, \ldots, N$ and finding the optimal rate allocation $R_l, l = 1, 2, \ldots, N$. It is shown in [58] that these two steps are separable, i.e., one can first find the optimal paths $\mathcal{P}^{l*}$ and then find the optimal rate allocation based on the optimal paths $\mathcal{P}^{l*}$. This separation holds even if the function $F(\cdot)$ is nonlinear. It is easy to see the optimal path $\mathcal{P}^{l*}$ is the path with minimum total weight. Then, the optimal routing structure is the shortest path tree rooted at the sink, which can be found effectively and in a distributed manner. Now, suppose that the cost function F is such that $F(R_l) = R_l$. In this case, the problem becomes

$$\min \sum_{l=1}^{N} R_l d_{\text{SPT}}(l,t)$$
$$s.t.(R_1, R_2, \ldots, R_N) \in SW_N$$

where $d_{\text{SPT}}(l,t)$ (known as a constant) is the weight of the path from source l to terminal t on the shortest path tree. This is a linear programming problem with number of constraints exponentially with N. However, because of the contra-polymatroidal structure of the Slepian–Wolf region [59], the solution can be found in a easy greedy manner as follows [58].

1. Find a permutation π such that $d_{\text{SPT}}(\pi(1),t) \geq d_{\text{SPT}}(\pi(2),t) \geq \cdots \geq d_{\text{SPT}}(\pi(N),t)$.
2. The optimal rate allocations is given by

$$\begin{aligned} R_{\pi(1)} &= H(X_{\{\pi(1)\}} | X_{\{\pi(2),\pi(3),\ldots,\pi(N)\}}) \\ R_{\pi(2)} &= H(X_{\{\pi(2)\}} | X_{\{\pi(3),\pi(4),\ldots,\pi(N)\}}) \\ &\vdots \\ R_{\pi(N)} &= H(X_{\{\pi(N)\}}) \end{aligned} \tag{7.7}$$

If the function $F(\cdot)$ is not linear but convex, the problem can still be solved by convex optimization [51] but the simple greedy algorithm may not work here.

From the previous discussion, we know that Slepian–Wolf coding along with routing is the optimal solution for the single sink data collection problem. In fact, it is shown in [60] that in terms of the cost under convex and increasing cost functions, Slepian–Wolf coding plus routing is still the optimal solution even if the wireless network broadcast advantage is considered. Interestingly, because the N-dimensional ($N > 2$) Slepian–Wolf code design problem remains open, [58, 60] also consider several schemes that do not use distributed source coding but allow some cooperation among the sources. Clearly, the communication between the sources will increase the cost. The cost of the Hierarchical Difference Broadcasting in [60] has been shown to have the same order compared to Slepian–Wolf coding. However, the explicit communication scheme in [58] will have significant loss compared to Slepian–Wolf under some conditions.

7.5 Networked Distributed Source Coding: Multiple Terminals

We now consider the variant of the problem when there are multiple terminals that want to reconstruct all the sources. This is called multicast. As before, one could attempt to treat this scenario as a generalization of the single terminal case. For example, one could divide the capacity of each edge into various parts, with each part responsible for conveying the bits to a specific terminal. However, on closer inspection it is possible to realize that such a strategy will in general be suboptimal.

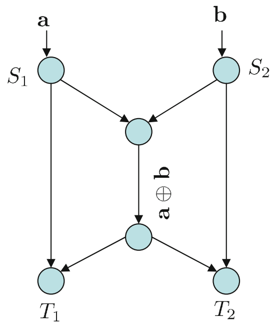


Fig. 7.6 A network with unit capacity edges and sources S_1 and S_2 and terminals T_1 and T_2. Each terminal wants to simultaneously recover the pair of bits (a, b). Under routing this is impossible. However, by computing and sending $a \oplus b$ along the bottleneck edge, we can achieve simultaneous recovery

To see this consider Fig. 7.6 that depicts the celebrated butterfly network of network coding [2]. In this example, each edge has unit capacity. Each terminal seeks to obtain the bits from both the sources. It is easy to see that if we only allow routing in the network, it is impossible to support this since the edge in the middle is a bottleneck. However, if we allow coding at the intermediate nodes and transmit the XOR of the two bits, then both terminals can obtain the two bits by simple XOR decoding. This example shows the potential gain of coding when there are multiple terminals. Of course, in this case the sources are independent. However, since independence is a degenerate case of correlation, one expects that similar conclusions will hold in the correlated case. As we shall see this is indeed the case. Furthermore, several interesting conclusions about the relationship of the coding rates and flow structures can be found.

7.5.1 A network Coding Primer

Traditionally, the intermediate nodes (routers) in the network only copy and forward packets. In a single source single sink unicast connection, routing achieves maximum flow, which equals to the minimum cut between the source and the terminal [61]. However, in a multicast scenario, pure routing may not achieve maximum flow as shown above. But it has been shown in [2] that network coding achieves max-flow min-cut upper bound in multicast. Next, we shall mathematically describe this result.

As usual, we model a network as a graph $G = (V, E, C)$, where $C = \{c_e : e \in E\}$ is the capacity of the edges, where c_e is the capacity on edge e. The seminal work on network coding [2] finds a tight capacity characterization for the single source, multiple terminals multicast problem.

Theorem 2 *Consider a network $G = (V, E, C)$ with source s and L terminals: $t_1, \ldots, t_L$. Suppose that the source node observes a source X, such that its entropy $H(X) = R$. Each terminal can recover X if and only if*

$$min - cut(s, t_l) \geq R, \forall l \in \{1, \ldots, L\} \tag{7.8}$$

The work of [62, 63] shows that the multicast can be supported even with linear codes. Basically, each intermediate node transmits linear combinations of the packets, where a packet is treated as a vector over a finite field. It is possible to show that in this case at each terminal, the received packets are the source messages multiplied by a transfer matrix. By inverting the transfer matrix, the terminal is able to recover the source packets. Moreover, as long as the coefficients of the linear combinations are chosen randomly from a large field and the min-cut between the source and each destination is greater than the source rate, the probability that the transfer matrix is invertible is very high [64]. This fact provides a simple distributed scheme for network coding-based multicast. A practical multicast protocol based on these ideas was developed in [65].

7.5.2 Multicasting Correlated Sources over a Network

In the discussion in the previous section, we only considered multicast with single source. The multiple independent sources case can be reduced to single source case [63], by introducing a virtual supersource that is connected to each source node.

In this section we consider the problem of multicasting correlated sources over a network. We begin by stating the main result. Consider a network $G = (V, E, C)$, with terminals $t_i, i = 1, \ldots, L$ and a set of source node $\mathcal{S} \subset V$. Without loss of generality, we assume a numbering so that these are the first $|\mathcal{S}|$ sources in V. Furthermore, source node i observes a source X_i. The communication requirement for multicasting correlated sources is that each terminal $t_i, i = 1, \ldots, L$ needs to recover all sources $(X_1, \ldots, X_{|\mathcal{S}|})$ losslessly. The admissible rate region is given by [66, 67].

Theorem 3 *The correlated sources* $(X_1, \ldots, X_{|\mathcal{S}|})$ *can be multicast to the terminals* $t_1, \ldots, t_L$ *if and only if*

$$H(X_S|X_{S^c}) \leq \min -\text{cut}(S, t_i) \;\; \forall S \subseteq \mathcal{S} \tag{7.9}$$

An achievability scheme based on random linear network coding for this result was proposed in [64]. Alternative proofs are provided in [66, 67]. We briefly overview the achievability scheme in [64] now.

Consider two correlated sources generating binary vectors $\mathbf{x}_1, \mathbf{x}_2$ of length r_1 and r_2 according to joint probability distribution $Q(\mathbf{x}_1, \mathbf{x}_2)$ each time. After n time slots, the source messages are $\mathbf{x}_1^n$ and $\mathbf{x}_2^n$ of length nr_1 and nr_2, respectively. We assume that c_e is rational for all e. Furthermore assume that n is large enough so that $n \times c_e$ is an integer for all e. This implies that when considered over a block of n time slots we communicate nc_e bits over edge e.

Simply perform random linear coding at each node over a blocklength of n including the source nodes and intermediate nodes, i.e., the bits on an outgoing edge of node v are a linear combination of bits on incoming edges to node v, where the coefficients are chosen uniformly randomly from $GF(2)$. Each terminal t receives a vector $\mathbf{z}_t^n$ of length $n|\Gamma_i(t)|$, where $|\Gamma_i(t)|$ is the number of incoming edges to terminal t (before the edges are copied). Using the algebraic network coding framework [63], we can conclude that

$$\mathbf{z}_t^n = \left[\mathbf{x}_1^n \ \mathbf{x}_2^n\right] M_t \tag{7.10}$$

where M_t is a $(nr_1 + nr_2) \times n|\Gamma_i(t)|$ transfer matrix from the sources to terminal t. When sources are independent, M_t needs to have full rank so that by inversion we can recover the sources. In the case of correlated sources , M_t need not have full rank because we can take advantage of the correlation between the sources to find $\mathbf{x}_1^n$ and $\mathbf{x}_2^n$.

The decoding is done as follows. Find all possible $\left[\mathbf{x}_1^n \ \mathbf{x}_2^n\right]$ satisfying (7.10). Note that $\mathbf{x}_1^n, \mathbf{x}_2^n$ can be viewed as a length n vector of elements from $GF(2^{r_1})$ and $GF(2^{r_2})$, respectively.[5] Let $\mathbf{x}_{1i}, \mathbf{x}_{2i}$ denote the ith element and $i = 1, 2, \ldots, n$. The number of appearances of $(\mathbf{a}, \mathbf{b}), \mathbf{a} \in GF(2^{r_1}), \mathbf{b} \in GF(2^{r_2})$ is defined to be $N(\mathbf{a}, \mathbf{b}) = |\{i : \mathbf{x}_{1i} = \mathbf{a}, \mathbf{x}_{2i} = \mathbf{b}\}|$. The empirical joint distribution (histogram) $P_{\mathbf{x}_1^n, \mathbf{x}_2^n}$ is $P_{\mathbf{x}_1^n, \mathbf{x}_2^n}(\mathbf{a}, \mathbf{b}) = N(\mathbf{a}, \mathbf{b})/n$ for $\mathbf{a} \in GF(2^{r_1})$ and $\mathbf{b} \in GF(2^{r_2})$. The empirical joint distribution is an approximation of the true joint distribution based on the observation of two sequences $\mathbf{x}_1^n$ and $\mathbf{x}_2^n$. Note that the empirical joint distribution defined for each sequence $\left[\mathbf{x}_1^n, \mathbf{x}_2^n\right]$ has a similar form to a probability mass function. Then, the functions applied on probability mass function, such as entropy function, relative entropy function, can be applied to $P_{\mathbf{x}_1^n, \mathbf{x}_2^n}$.

In the decision procedure, given all sequences $\left[\mathbf{x}_1^n, \mathbf{x}_2^n\right]$ that satisfying $\mathbf{z}_t^n = \left[\mathbf{x}_1^n \mathbf{x}_2^n\right] M_t$, find

$$\left\{\hat{\mathbf{x}}_1^n, \hat{\mathbf{x}}_2^n\right\} = \arg \min_{\left[\mathbf{x}_1^n \mathbf{x}_2^n\right] M_t = \mathbf{z}_t^n} \alpha\left(P_{\mathbf{x}_1^n, \mathbf{x}_2^n}\right)$$

where $\alpha(\cdot)$ is a function that needs to be chosen, depending on the metric to be optimized. The two functions discussed below both achieve the capacity region in Theorem 3.

1. *Maximum-Q probability decoder.* $\alpha\left(P_{\mathbf{x}_1^n, \mathbf{x}_2^n}\right) = D\left(P_{\mathbf{x}_1^n, \mathbf{x}_2^n} || Q\right) + H\left(P_{\mathbf{x}_1^n, \mathbf{x}_2^n}\right)$, where $D(\cdot||\cdot)$ is the relative entropy [6],

$$D\left(P_{\mathbf{x}_1^n, \mathbf{x}_2^n} || Q\right) = \sum_{\mathbf{a} \in F_{2^{r_1}}} \sum_{\mathbf{b} \in F_{2^{r_2}}} P_{\mathbf{x}_1^n, \mathbf{x}_2^n}(\mathbf{a}, \mathbf{b}) \log \frac{P_{\mathbf{x}_1^n, \mathbf{x}_2^n}(\mathbf{a}, \mathbf{b})}{Q(\mathbf{a}, \mathbf{b})}$$

and $H(\cdot)$ is the joint entropy function [6]

[5] A length r vector with elements from $GF(2)$ can be viewed as an element from $GF(2^r)$.

$$H\left(P_{\mathbf{x}_1^n,\mathbf{x}_2^n}\right) = -\sum_{\mathbf{a}\in F_{2^{r_1}}}\sum_{\mathbf{b}\in F_{2^{r_2}}} P_{\mathbf{x}_1^n,\mathbf{x}_2^n}(\mathbf{a},\mathbf{b}) \log P_{\mathbf{x}_1^n,\mathbf{x}_2^n}(\mathbf{a},\mathbf{b})$$

From [6, theorem 12.1.2], since $\left(\mathbf{x}_1^n, \mathbf{x}_2^n\right)$ are drawn i.i.d. according to $Q(\mathbf{x}_1, \mathbf{x}_2)$, the probability of $\mathbf{x}_1^n, \mathbf{x}_2^n$ is given by

$$Q^n(\mathbf{x}_1, \mathbf{x}_2) = 2^{-n(D(P_{\mathbf{x}_1,\mathbf{x}_2}||Q)+H(P_{\mathbf{x}_1,\mathbf{x}_2}))}$$

Therefore, finding $\mathbf{x}_1^n, \mathbf{x}_2^n$ that minimizing $\alpha\left(P_{\mathbf{x}_1^n,\mathbf{x}_2^n}\right)$ is equivalent to finding $\mathbf{x}_1^n, \mathbf{x}_2^n$ that maximizing the sequence probability.

2. *Minimum entropy decoder.* $\alpha(P_{\mathbf{x}_1,\mathbf{x}_2}) = H(P_{\mathbf{x}_1,\mathbf{x}_2})$.
   Note that here the decoder does not need to know the prior source joint distribution Q. Thus, it is an universal decoder. For a long sequence, the empirical distribution $P_{\mathbf{x}_1,\mathbf{x}_2}$ is very close to the true distribution Q, which causes $D(P_{\mathbf{x}_1,\mathbf{x}_2}||Q)$ to approach zero. Therefore, the minimum entropy decoder is an approximation of maximum-Q probability decoder.

It is shown in [64] that as long as

$$min - cut(s_1, t_i) \geq H(X_1|X_2) \tag{7.11}$$

$$min - cut(s_2, t_i) \geq H(X_2|X_1) \text{ and} \tag{7.12}$$

$$min - cut(s_1 \text{ and } s_2, t_i) \geq H(X_1, X_2) \tag{7.13}$$

for every $i = 1, 2, \ldots, L$, each terminal t_i can recover X_1 and X_2 with vanishing error probability when the one of the two decoders shown above is used. Therefore, the admissible rate region achieves bound (7.9). However, the decoding algorithms above are based on exhaustive search and have a complexity that is unacceptably high.

7.5.3 Separating Distributed Source Coding and Network Coding

The achievability scheme described in the previous section performs distributed source coding and network coding jointly and has high decoding complexity. Perhaps the simplest way to multicast correlated sources is to perform distributed source coding and network coding separately, i.e., the source nodes perform distributed source coding (Slepian–Wolf coding) and the coded bits are multicasted to the terminals through network coding. The terminals first decode the network code to obtain the Slepian–Wolf coded bits, then jointly decode the Slepian–Wolf code (usually is a channel code) to recover the sources. The decoding algorithms for network code and Slepian–Wolf code have been well studied and have polynomial time complexity. However, the separation of distributed source coding and network coding is suboptimal in general [68].

At an intuitive level, this result can be understood as follows. Suppose that the network is such that each terminal can operate at the same point in the Slepian–Wolf region. In such a situation, one could use a Slepian–Wolf code and encode each source. Next, one could treat the encoded sources as independent and multicast the encoded bits to each terminal. The terminal then decodes to obtain the original sources. Roughly speaking, in this case we can reduce the correlated sources multicast to an independent sources multicast.

However, if different terminals in the network are forced to operate at different rate points in the Slepian–Wolf region, because of the nature of their connectivity, then a reduction to the independent sources multicast is not possible in general. In this case, clearly one cannot work with a single distributed source code. It can be shown that there exist instances of networks and source distributions such that performing separate distributed source coding and network coding can be strictly suboptimal with respect to the approach in [64]. A surprising conclusion of [68] is that if there are two sources and two terminals in a network, then it can be shown that there is no loss in using a separation-based approach. This result forms the basis of practical approaches to combining distributed source coding and network coding as explained in the next section.

7.5.4 Practical Joint Distributed Source Coding and Network Coding

In this section, we describe practical algorithms to perform joint distributed source coding and network coding. Suppose there are two source nodes $s_1, s_2 \in V$ and observe two binary sources X and Y, respectively. The sources generate bits i.i.d. according to the joint distribution $p(X, Y)$ where the joint distribution satisfies the following symmetry property, i.e., $p(X + Y = 1) = p < 0.5$. Then, as discussed before, the sequences $\mathbf{x}, \mathbf{y}$ are related by $\mathbf{y} = \mathbf{x} + \mathbf{e}$, where e_i equals to 1 with probability $p < 0.5$. Note that $H(X, Y) = 1+H_b(p)$ and $I(X; Y) = 1-H_b(p)$. Let H be the parity check matrix for a channel code approaching the capacity of a binary symmetric channel with crossover probability p with code rate $k/n = I(X; Y) = 1 - H_b(p)$, i.e., there is a decoding function $f(\cdot)$ such that $Pr(\mathbf{e} \neq f(\mathbf{e}H^T))$ is arbitrarily close to zero.

The basic idea is to transmit $\mathbf{x}H^T + \mathbf{y}H^T = \mathbf{e}H^T$ to each terminal such that $\mathbf{e}$ can be recovered. Then, we transmit some additional information so that each terminal can recover either $\mathbf{x}$ or $\mathbf{y}$. We shall see the exact form of this additional information later. The simplest but not necessarily optimal way to convey the sum $\mathbf{e}H^T = \mathbf{x}H^T + \mathbf{y}H^T$ to the terminal is to multicast both $\mathbf{x}H^T$ and $\mathbf{y}H^T$ to each terminal and compute the sum at the terminal. Based on this, a practical joint distributed source coding and network coding is proposed in [69]. We first describe this scheme and then discuss the optimal schemes to multicast the sum to the terminals. The scheme in [69] is not optimal in the sense that in general, it requires more network capacity than the result of [64] requires.

The design scheme can be summarized as follows. The network capacity resource C is partitioned into two shares: C_1 and C_2, where $C_1 + C_2 \leq C$. Each share is used to support two multicast sessions. Let $\bar{H}$ be a matrix such that $[\bar{H}^T H^T]$ has full rank. And let $\mathbf{x}_1 = \mathbf{x}\bar{H}, \mathbf{y}_1 = \mathbf{y}\bar{H}$. The two multicast sessions are described as follows.

1. In the first session, multicast $\mathbf{x}H^T$ and $\mathbf{y}H^T$ to each terminal. This implies $\mathbf{e}H^T$ can be computed, and $\mathbf{e}$ can be recovered at each terminal since H is the parity check matrix of a capacity achieving code. Using this, $\mathbf{e}_1 \equiv \mathbf{y}_1 + \mathbf{x}_1 = \mathbf{e}\bar{H}^T$ can be computed.
   The length of $\mathbf{x}H^T$ is $(n - k) = nH(X|Y)$ (likewise for $\mathbf{y}H^T$). We need to multicast $nH(X|Y)$ bits from node s_1 to the terminal and $nH(Y|X)$ bits from node s_2 to the terminal. This requires $G(V, E, C_1)$ to support a multicast with rate $H(X|Y) + H(Y|X)$ from a virtual supersource connected to s_1, s_2 to each terminal.
2. In the second session, the sources transmit linear combinations of $\mathbf{x}_1$ and $\mathbf{y}_1$ to the network and $\mathbf{x}_1 A_t + \mathbf{y}_1 B_t$ is received by terminal t. A_t and B_t are transfer matrices from s_1 to terminal t and s_2 to terminal t, respectively, and they are assumed known to the terminal t. A_t and B_t are such that given $\mathbf{e}_1$ and $\mathbf{x}_1 A_t + \mathbf{y}_1 B_t$, $\mathbf{x}_1, \mathbf{y}_1$ can be recovered. Since we can compute $(\mathbf{x}_1 + \mathbf{y}_1)B_t = \mathbf{e}_1 B_t = \mathbf{e}\bar{H}^T B_t$ and then $\mathbf{x}_1(A_t + B_t) = \mathbf{x}_1 A_t + \mathbf{y}_1 B_t + \mathbf{e}_1 B_t$, as long as $A_t + B_t$ is invertible, $\mathbf{x}_1$ and $\mathbf{y}_1$ can be recovered. The invertibility of $A_t + B_t$ is guaranteed with high probability (for details see [69]). After $\mathbf{x}_1$ is obtained, we compute $\mathbf{y}_1 = \mathbf{e}_1 + \mathbf{x}_1$. Once $\mathbf{x}_1, \mathbf{y}_1$ are known, $\mathbf{x}, \mathbf{y}$ can be recovered by the inversion of $[\bar{H}^T H^T]$ since $[\mathbf{x}\bar{H}^T \mathbf{x}H^T] = \mathbf{x}[\bar{H}^T H^T]$ and $\mathbf{y} = \mathbf{x} + \mathbf{e}$.

The two multicast sessions are illustrated in Fig. 7.7. The admissible rate region for this design scheme is

$$\mathbf{C}^* = \{C_1 + C_2 : C_1 \in \mathbf{C}(\bar{s}, T, H(X|Y) + H(Y|X)) \text{ and } C_2 \in \mathbf{C}(u, T, I(X;Y))\}$$

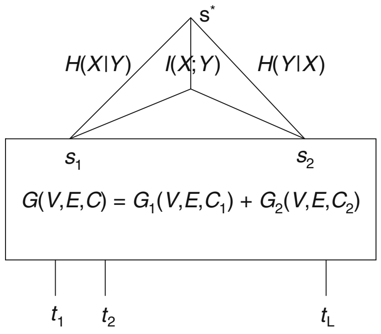


Fig. 7.7 Multicast model for the practical scheme [69]

In general, $\mathbf{C}^*$ requires more capacity than the optimal capacity region [64] because separate multicast sessions are usually suboptimal. But if there are only two terminals (and we are only dealing with two sources), $\mathbf{C}^*$ is optimal, i.e., the practical design scheme is optimal [68, 69].

Computing the sum at the terminals (see [70–72] for related work) may not be optimal in terms of the capacity region. It may in fact be better in terms of resource utilization if the sum is computed at some intermediate nodes and then sent to each terminal. In a network with two sources multiple terminals or two terminals multiple sources, it is shown in [70, 72] that the optimal scheme to convey the symbol sum of the sources to the terminals is to compute the sum in some intermediate nodes and multicast to the terminals. In general, finding the right set of nodes at which the sum should be computed is an open problem. But, the idea of computing the sum at the intermediate nodes leads us to a heuristic approach to the joint distributed source coding and network coding. We can find a set of nodes U and multicast $\mathbf{x}H^T$ and $\mathbf{y}H^T$ to each node in U (*multicast session 1*). Then, compute the sum $\mathbf{e}H^T$ at $u \in U$ and multicast to the terminals so that each terminal can recover $\mathbf{e}$ (*multicast session 2*). Transmit linear combinations $\mathbf{x}A_t + \mathbf{y}B_t$ to the terminals (*multicast session 3*) and if $(A_t + B_t)$ is invertible then both $\mathbf{x}$ and $\mathbf{y}$ can be recovered in a similar manner to the previous scheme. Note that the coded packets in *multicast session 1* can be used in *multicast session 3* since $\mathbf{x}H^T$ and $\mathbf{y}H^T$ are also linear combinations of $\mathbf{x}$ and $\mathbf{y}$. Next we demonstrate an example of this scheme in which we achieve the optimal capacity region.

Consider the network in Fig. 7.8. The capacity on each edge is 0.5. The source nodes s_1, s_2 observe the sources X and Y, and they are correlated such that $H(X) = H(Y) = 1$ and $H(Y|X) = H(X|Y) = 0.5$. The terminals are t_1, t_2, and t_3

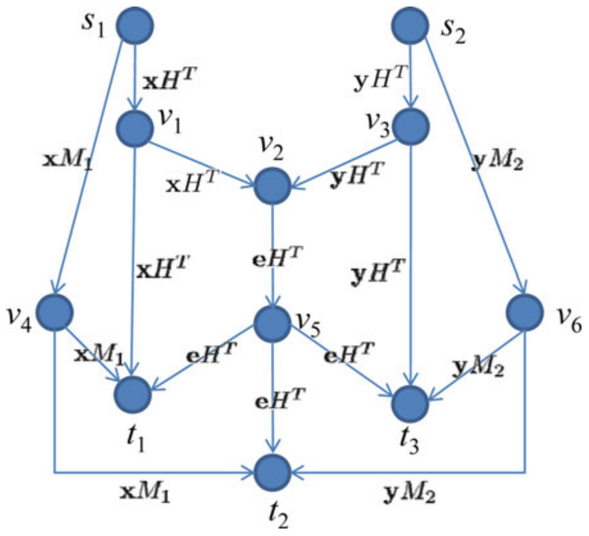


Fig. 7.8 An example where the strategy in [69] is suboptimal. However, our proposed heuristic for selecting the right set of nodes for computing the sum works better

and $\min-\text{cut}(s_i, t_j) = 0.5$ for $i = 1, 2$, $j = 1, 2, 3$, $\min-\text{cut}(\{s_1, s_2\}, t_i) = 1.5$ for $i = 1, 2, 3$. According to Theorem 3, the capacity of this network supports the recovery of the sources at the terminals. Consider the following scheme: s_1, s_2 transmit $\mathbf{x}H^T$ and $\mathbf{y}H^T$ to node v_2 (*multicast session 1*). Node v_2 computes the sum $\mathbf{e}H^T$ and routes it to the terminals (*multicast session 2*). For *multicast session 3*, transmit $\mathbf{x}H^T$, $\mathbf{y}H^T$ on $v_1 - t_1$, $v_3 - t_3$, respectively.[6] In addition, s_1, s_2 transmit random linear combinations on edges $s_1 - v_4$, $s_2 - v_6$, i.e., M_1, M_2 are matrices of dimension $n \times 0.5n$ consisting of entries randomly from $GF(2)$. Then, matrices M_1, M_2, and $[M_1 M_2]$ have full rank with high probability. Terminal t_1 receives $\mathbf{e}H^T$, $\mathbf{x}M_1$ and $\mathbf{x}H^T$. From the first one t_1 can decode $\mathbf{e}$ and from the last two t_1 can recover $\mathbf{x}$, then $\mathbf{y} = \mathbf{x} + \mathbf{e}$ can also be obtained. Terminal t_2 acts in a similar fashion as t_1, while t_3 can decode $\mathbf{e}$ from $\mathbf{e}H^T$ and it also knows $\mathbf{x}M_1$ and $\mathbf{y}M_2$. Therefore, it can compute $\mathbf{x}M_2 = \mathbf{e}M_2 + \mathbf{y}M_2$ then $\mathbf{x}$ can be recovered by the inversion of $[M_1 M_2]$.

As shown in [69], the scheme that multicasts both $\mathbf{x}H^T$ and $\mathbf{y}H^T$ to the terminals cannot achieve the capacity region in the example above. But from some simulations on random graphs, where the optimal set U is found by integer programming, we observe that in many cases, multicasting both $\mathbf{x}H^T$ and $\mathbf{y}H^T$ to the terminals and computing the sum there is as good as computing the sum at some intermediate nodes. Clearly, the best choice of nodes for computing the sum depends on the network topology. The problem of choosing these nodes in an efficient manner is still an open problem.

7.5.5 Resource Allocation for Multicasting Correlated Sources over a Network

Given the admissible region in Sect. 7.5.2, it is natural question to determine the rate at each source and the flow on each edge such that the total cost is minimized. The problem is solved in an efficient manner in [73, 74].

The network is modeled as a directed acyclic graph $G = (V, E, C)$ and each edge is associated with a weight w_{ij}. For simplicity we assume that the cost of the use of an edge (i, j) when the actual data rate on edge (i, j) is z_{ij} is $w_{ij}z_{ij}$. To facilitate the problem formulation we append a virtual super source node s^* to G, so that

$$\begin{aligned}
V^* &= V \cup \{s^*\} \\
E^* &= \{(s^*, v) |\ v \in S\} \cup E \text{ and} \\
C^*_{ij} &= \begin{cases} C_{ij} & (i, j) \in E \\ H(X_j) & \text{if } i = s^* \text{ and } j \in S \end{cases}
\end{aligned}$$

We let $G^* = (V^*, E^*, C^*)$. Denote the source node set as $\mathcal{S}$ and the terminal set as T. The admissible region in Theorem 3 requires the min-cut between any

[6] We could also simply perform random linear network coding on these edges.

subset S of nodes and *every* terminal greater than $H(S|S^c)$. From max-flow min-cut theorem, we know the min-cut can be characterized as max-flow. As long as there is a flow of value R from a source s to a terminal t, the min-cut between s and t is R. Thus, to model the conditions on the min-cut, we introduce virtual flows $\mathbf{f}^{(t_k)} = \left\{ f_{ij}^{(t_k)} \right\}$ for each terminal t_k. Note that we only require the existence of the flow for every terminal; the flows corresponding to different terminals can coexist on an edge. So the actual flow rate z_{ij} on edge (i, j) is the maximum (not the sum) of $f_{ij}^{(t_k)}, \forall t_k \in T$, i.e., $z_{ij} \geq f_{ij}^{(t_k)}, \forall t_k \in T$. Based on the discussions above, the problem can be formulated as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in E} w_{ij} z_{ij} \\
\text{s. t.} \quad & 0 \leq f_{ij}^{(t_k)} \leq z_{ij} \leq C_{ij}^*, \quad (i, j) \in E^*, t_k \in T \\
& \sum_{\{j|(i,j)\in E^*\}} f_{ij}^{(t_k)} - \sum_{\{j|(j,i)\in E^*\}} f_{ji}^{(t_k)} = \sigma_i^{(t_k)}, \text{for } i \in V^*, t_k \in T, \quad (7.14) \\
& f_{s^*i}^{(t_k)} \geq R_i^{(t_k)}, \quad \text{for } i \in \mathcal{S}, t_k \in T \quad (7.15) \\
& \left(R_1^{(t_k)}, R_2^{(t_k)}, \ldots, R_N^{(t_k)} \right) \in SW_N, \quad \text{for } t_k \in T \quad (7.16)
\end{aligned}$$

where

$$\sigma_i^{(t_k)} = \begin{cases} H(X_1, X_2, \ldots, X_N) & \text{if } i = s^* \\ -H(X_1, X_2, \ldots, X_N) & \text{if } i = t_k \\ 0 & \text{otherwise} \end{cases}$$

The constraint (7.14) is the flow balance constraint for each virtual flow. The constraints (7.15) and (7.16) make sure for each terminal t_k there is a flow of value $H(X_S|X_{S^c})$ from each subset S of sources to t_k. The detailed proof of the correctness of the formulation can be found in [73, 74]. The formulation of *MIN-COST-SW-NETWORK* as presented above is a linear program and can potentially be solved by a regular LP solver. However, the number of constraints due to the requirement that $\mathbf{R} \in \mathcal{SW}_\mathcal{N}$ is $|T|(2^N - 1)$ that grows exponentially with the number of sources. For regular LP solvers the time complexity scales with the number of constraints and variables. Thus, using a regular LP solver is certainly not time efficient. Moreover even storing the constraints consumes exponential space and thus using a regular LP solver would also be space inefficient. We now present efficient techniques for solving this problem.

Let $\mathbf{w}, \mathbf{z}, \mathbf{f}^{(t_k)}$ denote the column vectors of $w_{ij}, z_{ij}, f_{ij}^{(t_k)}$ for $(i, j) \in E$ and $\mathbf{R}^{(t_k)}, \mathbf{f}_{s*}^{(t_k)}$ denote the column vectors of $R_i^{(t_k)}, f_{s*i}^{(t_k)}$ for $i = 1, 2, \ldots, |\mathcal{S}|$. Let L be the number of terminals. We form the Lagrangian of the optimization problem with respect to the constraints $\mathbf{R}^{(t_k)} \leq \mathbf{f}_{s*}^{(t_k)}$, for $t_k \in T$. This is given by

$$\begin{aligned}
&L(\lambda, \mathbf{z}, \mathbf{f}^{(t_1)}, \ldots, \mathbf{f}^{(t_L)}, \mathbf{R}^{(t_1)}, \ldots, \mathbf{R}^{(t_L)}) \\
&= \mathbf{w}^T \mathbf{z} + \sum_{k=1}^{L} \lambda_k^T \left(\mathbf{R}^{(t_k)} - \mathbf{f}_{s^*}^{(t_k)} \right)
\end{aligned}$$

where $\lambda = \left[\lambda_1^T \; \lambda_2^T \; \ldots \; \lambda_L^T\right]^T$ and $\lambda_k = [\lambda_{k,1}, \lambda_{k,2}, \ldots, \lambda_{k,|\mathcal{S}|}]^T$ are the dual variables such that $\lambda \succeq 0$ (where $\succeq$ denotes component-wise inequality).

For a given λ, let $g(\lambda)$ denote the dual function obtained by

$$g(\lambda) = \text{minimize}_{\mathbf{z}, \mathbf{f}^{(t_1)}, \ldots, \mathbf{f}^{(t_L)}, \mathbf{R}^{(t_1)}, \ldots, \mathbf{R}^{(t_L)}} L\left(\lambda, \mathbf{z}, \mathbf{f}^{(t_1)}, \ldots, \mathbf{f}^{(t_L)}, \mathbf{R}^{(t_1)}, \ldots, \mathbf{R}^{(t_L)}\right)$$

Since strong duality holds in our problem we are guaranteed that the optimal value of *MIN-COST-SW-NETWORK* can be equivalently found by maximizing $g(\lambda)$ subject to $\lambda \succeq 0$ [51]. Thus, if $g(\lambda)$ can be determined in an efficient manner for a given λ then we can hope to solve *MIN-COST-SW-NETWORK* efficiently.

Consider the optimization problem for a given $\lambda \succeq 0$.

$$\begin{aligned}
\text{minimize} \quad & \mathbf{w}^T \mathbf{z} + \sum_{k=1}^{L} \lambda_k^T \left(\mathbf{R}^{(t_k)} - \mathbf{f}_{s^*}^{(t_k)} \right) \\
\text{s. t.} \quad & 0 \le f_{ij}^{(T_k)} \le z_{ij} \le C_{ij}, \quad (i,j) \in E^*, t_k \in T \\
& \sum_{\{j|(i,j) \in E^*\}} f_{ij}^{(t_k)} - \sum_{\{j|(j,i) \in E^*\}} f_{ji}^{(t_k)} = \sigma_i^{(t_k)}, \; i \in V^*, t_k \in T \\
& \mathbf{R}^{(t_k)} \in SW_N, \; t_k \in T
\end{aligned} \tag{7.17}$$

We realize on inspection that this minimization decomposes into a set of independent subproblems shown below.

$$\begin{aligned}
\text{minimize} \quad & \mathbf{w}^T \mathbf{f} - \sum_{k=1}^{L} \lambda_k^T \mathbf{f}_{s^*}^{(t_k)} \\
\text{s. t.} \quad & 0 \le f_{ij}^{(t_k)} \le z_{ij} \le C_{ij}, \quad (i,j) \in E^*, t_k \in T \\
& \sum_{\{j|(i,j) \in E^*\}} f_{ij}^{(t_k)} - \sum_{\{j|(j,i) \in E^*\}} f_{ji}^{(t_k)} = \sigma_i^{(t_k)}, \; i \in V^*, t_k \in T
\end{aligned} \tag{7.18}$$

and for each $t_k \in T$,

$$\begin{aligned}
\text{minimize} \quad & \lambda_k^T \mathbf{R}^{(t_k)} \\
\text{subject to} \quad & \mathbf{R}^{(t_k)} \in SW_N
\end{aligned} \tag{7.19}$$

The optimization problem in (7.18) is a linear program with variables z and $x^{(T_k)}$ for $k = 1, \ldots, N_R$ and a total of $(2|T| + 1)|E^*| + |T||V^*|$ constraints that can be solved efficiently by using a regular LP solver. It can also be solved by treating it as a minimum cost network flow problem with fixed rates for which many efficient techniques have been developed [50].

However, each of the subproblems in (7.19) still has $2^N - 1$ constraints and therefore the complexity of using an LP solver is still exponential in N. However, recall the contra-polymatroid property of Slepian–Wolf region mentioned in Sect. 7.4.1.2. Using the contra-polymatroid property, the solution to this LP can be found by a greedy allocation of the rates as shown in (7.7), where the permutation π is such that $\lambda_{k,\pi(1)} \geq \lambda_{k,\pi(2)} \geq \cdots \geq \lambda_{k,\pi(N)}$.

The previous algorithm presents us a technique for finding the value of $g(\lambda)$ efficiently. It remains to solve the maximization

$$\max_{\lambda \succeq 0} g(\lambda)$$

For this purpose we use the fact that the dual function is concave (possibly non-differentiable) and can therefore be maximized by using the projected subgradient algorithm [75]. Roughly speaking, the subgradient algorithm is a iterative method to minimize non-differentiable convex (or maximize concave) functions. It is similar to the gradient descent method, though there are notable differences. The subgradient for λ_k can be found as $\mathbf{R}^{(t_k)} - \mathbf{f}_{s*}^{(t_k)}$ [75].

Let λ^i represent the value of the dual variable λ at the ith iteration and θ_i be the step size at the ith iteration. A step-by-step algorithm to solve *MIN-COST-SW-NETWORK* is presented below.

1. Initialize $\lambda^0 \succeq 0$.
2. For given λ^i solve the problem (7.18) using an LP solver and for each $t_k \in T$, solve the problem (7.19) using the greedy algorithm presented in (7.7).
3. Set $\lambda_k^{i+1} = \left[\lambda_k^i + \theta_i \left(\mathbf{R}^{(t_k)} - \mathbf{f}_{s*}^{(t_k)}\right)\right]^+$ for all $t_k \in T$, where $[x]^+ = x$ if $x \geq 0$ and zero otherwise. Goto step 2 and repeat until convergence.

While subgradient optimization provides a good approximation on the optimal value of the primal problem, a primal-optimal solution or even a feasible, near-optimal solution is usually not available because the objective function is linear. In our problem, we seek to jointly find the flows and the rate allocations that support the recovery of the sources at the terminals at minimum cost. Thus, finding the appropriate flows and rates specified by the primal-optimal or near primal-optimal $\mathbf{z}, \mathbf{f}^{(t_1)}, \ldots, \mathbf{f}^{(t_L)}, \mathbf{R}^{(t_1)}, \ldots, \mathbf{R}^{(t_L)}$ is important. Toward this end we use the method of Sherali and Choi [76]. We skip the details and refer the interested reader to [73, 74].

7.6 Conclusion

In this survey we have examined the problem of distributed source coding over networks. Distributed source coding has been traditionally studied under a model where there exist direct source destination links. In a general network, the sources communicate with the destinations over a network whose topology may be quite complicated. It turns out that in this case the problem of distributed source coding and network information transfer needs to be addressed jointly. In particular, treating these problems separately can be shown to be suboptimal in general. Moreover, in certain cases the usage of the network coding [2] becomes essential. We also discussed various resource allocation problems that occur in this space and provided an overview of the solution approaches.

There are several problems that need to be addressed in this area. In the area of sensor networks, it would be interesting to examine if simple protocols can be developed that leverage joint distributed source coding and network coding. In this survey we assumed that the source statistics are known to the intended destination. In practice, the protocols will need to ensure that these statistics are communicated periodically. In a practical sensor network, it is reasonable to assume that some limited communication between the sensors is possible. It would be interesting to see if this reduces the overall complexity of decoding at the destinations.

Acknowledgments This work was supported in part by NSF grant CNS-0721453.

References

1. D. Slepian and J. Wolf. Noiseless coding of correlated information sources. *IEEE Transactions on Information Theory*, 19:471–480, July 1973.
2. R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.
3. X. Lin and N. Shroff. Utility maximization for communication networks with multipath routing. *IEEE Transactions on Automatic Control*, 51(5):766–781, May 2006.
4. M. Chiang, S. Low, A. Calderbank, and J. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, January 2007.
5. C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.
6. T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, New York, NY, 1991.
7. Z. Xiong, A. Liveris, and S. Cheng. Distributed source coding for sensor networks. *Signal Processing Magazine, IEEE*, 21(5):80–94, September 2004.
8. T. Cover. A proof of the data compression theorem of slepian and wolf for ergodic sources (corresp.), *IEEE Transactions on Information Theory*, 21(2):226–228, March 1975.
9. V. Stankovic, A. Liveris, Z. Xiong, and C. Georghiades. On code design for the slepian-wolf problem and lossless multiterminal networks. *IEEE Transactions on Information Theory*, 52(4):1495–1507, April 2006.
10. C.-F. Lan, A. Liveris, K. Narayanan, Z. Xiong, and C. Georghiades. Slepian-wolf coding of multiple m-ary sources using ldpc codes. In: *Data Compression Conference, 2004. Proceedings. DCC 2004*, page 549, March 2004.
11. A. Liveris, C. Lan, K. Narayanan, Z. Xiong, and C. Georghiades. Slepian-Wolf coding of three binary sources using LDPC codes. In: *Proceedings of International Symposium on Turbo Codes and Related Topics, Brest, France*, Sep. 2003.
12. A. Wyner. Recent results in Shannon theory. *IEEE Transactions on Information Theory*, 20:2–10, January 1974.

13. S. S. Pradhan and K. Ramchandran. Distributed Source Coding using Syndromes (DISCUS): Design and Construction. *IEEE Transactions on Information Theory*, 49:626–643, March 2003.
14. A. Liveris, Z. Xiong, and C. N. Georghiades. Compression of binary sources with side information at the decoder using LDPC codes. *IEEE Communications Letters*, 6(10):440–442, 2002.
15. D. Varodayan, A. Aaron, and B. Girod. Rate-adaptive codes for distributed source coding. *Signal Processing*, 86(11):3123–3130, 2006.
16. P. Tan and J. L. Tiffany. A general and optimal framework to achieve the entire rate region for slepian-wolf coding. *Signal Processing*, 86(11):3102–3114, 2006.
17. J. Muramatsu and T. Uyematsu, and T. Wadayama. Low-density parity-check matrices for coding of correlated sources. *IEEE Transactions on Information Theory*, 51(10):3645–3654, 2005.
18. M. Sartipi and F. Fekri. Distributed source coding in wireless sensor networks using LDPC coding: the entire Slepian-Wolf rate region. In: *IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, USA, March 2005.
19. S. Pradhan and K. Ramchandran. Distributed source coding: symmetric rates and applications to sensor networks. In: *Data Compression Conference, 2000. Proceedings. DCC 2000*, pages 363–372, 2000.
20. S. Lin and D. J. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, New Jersey, 2004.
21. R. Gallager. *Information Theory and Reliable Communication*, Wiley, New York, 1968.
22. T. Richardson and R. Urbanke. *Modern Coding Theory*, Cambridge University Press, Cambridge, 2008.
23. J. Garcia-Frias, Y. Zhao, and W. Zhong. Turbo-like codes for transmission of correlated sources over noisy channels. *Signal Processing Magazine, IEEE*, 24(5):58–66, September 2007.
24. J. Garcia-Frias and Y. Zhao. Compression of correlated binary sources using turbo codes. *IEEE Communications Letters*, 5(10):417–419, October 2006.
25. A. Aaron and B. Girod. Compression with side information using turbo codes. In: *Proceedings of IEEE Data Compression Conference(DCC)*, pages 252–261, 2002.
26. T. P. Coleman, A. H. Lee, M. Medard, and M. Effros. Low-complexity approaches to Slepian-Wolf near-lossless distributed data compression. *IEEE Transactions on Information Theory*, 52(8):3546–3561, 2006.
27. B. Rimoldi and C. Urbanke. Asynchronous Slepian-Wolf coding via source-splitting. In: *Proceedings of International Symposium on Information Theory, Ulm, Germany*, page 271, Ulm, Germany, June–July 1997.
28. S. Y. Tung, Multiterminal source coding, *Ph.D. Dissertation*, Cornell University, 1978.
29. K. Housewright. Source coding studies for multiterminal systems. *Ph.D. Dissertation*, University of California, Los Angeles, 1977.
30. T. Berger. Multiterminal source coding. In: *CISM Courses and Lectures No. 229, The Information Theory Approach to Communications*, Springer-Verlag, Berlin, 1980.
31. W. Kang and S. Ulukus. An outer bound for the multi-terminal rate-distortion region. In: *IEEE International Symposium on Information Theory*, pages 1419–1423, Seattle, Washington, USA, July 2006.
32. T. Berger. Multiterminal rate-distortion theory revisited. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference*, pages 1535–1537, September 2008.
33. Y. Oohama. Gaussian multiterminal source coding. *IEEE Transactions on Information Theory*, 43(6):1912–1923, November 1997.
34. Y. Oohama, Rate-distortion theory for gaussian multiterminal source coding systems with several side informations at the decoder. *IEEE Transactions on Information Theory*, 51(7):2577–2593, July 2005.

35. Y. Yang, V. Stankovic, Z. Xiong, and W. Zhao. On multiterminal source code design. In *Data Compression Conference, 2005. Proceedings. DCC 2005*, pages 43–52, Snowbird, Utah, USA, March 2005.
36. Y. Zhang, Z. Xiong, and Y. Yang. Code design for quadratic gaussian multiterminal source coding: The symmetric case. In: *IEEE International Symposium on Information Theory*, 28:1458–1462, July 3, 2009.
37. A. Wyner and J. Ziv. The rate-distortion function for source coding with side information at the decoder. *IEEE Transactions on Information Theory*, 22(1):1–10, January 1976.
38. R. Zamir, S. Shamai, and U. Erez. Nested linear/lattice codes for structured multiterminal binning. *IEEE Transactions on Information Theory*, 48(6):1250–1276, June 2002.
39. S. Servetto, Lattice quantization with side information: Codes, asymptotics, and applications in sensor networks. *IEEE Transactions on Information Theory*, 53(2):714–731, February 2007.
40. Z. Liu, S. Cheng, A. Liveris, and Z. Xiong. Slepian-wolf coded nested quantization (swc-nq) for wyner-ziv coding: performance analysis and code design. In: *Data Compression Conference, 2004. Proceedings. DCC 2004*, pages 322–331, March 2004.
41. B. Girod, A. Aaron, S. Rane, and D. Rebollo-Monedero. Distributed video coding. *Proceedings of the IEEE*, 93(1):71–83, January 2005.
42. Y. Yang, S. Cheng, Z. Xiong, and W. Zhao. Wyner-ziv coding based on tcq and ldpc codes. *IEEE Transactions on Communications*, 57(2):376–387, February 2009.
43. T. Berger, Z. Zhang, and H. Viswanathan. The ceo problem [multiterminal source coding]. *IEEE Transactions on Information Theory*, 42(3):887–902, May 1996.
44. H. Viswanathan and T. Berger. The quadratic gaussian ceo problem. *IEEE Transactions on Information Theory*, 43(5):1549–1559, September 1997.
45. Y. Oohama. The rate-distortion function for the quadratic gaussian ceo problem. *IEEE Transactions on Information Theory*, 44(3):1057–1070, May 1998.
46. V. Prabhakaran, D. Tse, and K. Ramachandran. Rate region of the quadratic gaussian ceo problem. In: *IEEE International Symposium on Information Theory*, page 119, June-2 July 2004.
47. J.-H. Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(4):609–619, August 2004.
48. T. S. Han. Slepian-wolf-cover theorem for networks of channels. *Information and Control*, 47(1):67–83, October 1980.
49. J. Barros and S. Servetto. Network information flow with correlated sources. *IEEE Transactions on Information Theory*, (52):155–170, January 2006.
50. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, New Jersey, 1993.
51. S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, 2004.
52. A. Roumy and D. Gesbert, Optimal matching in wireless sensor networks. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):725–735, December 2007.
53. S. Li and A. Ramamoorthy. Rate and power allocation under the pairwise distributed source coding constraint. *IEEE Transactions on Communications*, 57(12), December 2009.
54. Y. J. Chu and T. H. Liu, On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965.
55. R. Cristescu and B. Beferull-Lozano. Lossy network correlated data gathering with high-resolution coding. *IEEE Transactions on Information Theory*, 52(6):2817–2824, June 2006.
56. R. Giles. Optimum matching forests I: Special weights. *Mathematical Programming*, 22(1):1–11, December 1982.
57. R. Giles, Optimum matching forests ii: General weights. *Mathematical Programming*, 22(1):12–38, December 1982.

58. R. Cristescu, B. Beferull-Lozano, and M. Vetterli. Networked slepian-wolf: theory, algorithms, and scaling laws. *IEEE Transactions on Information Theory*, 51(12):4057–4073, December 2005.
59. D. Tse and S. Hanly. Multiaccess fading channels. i. polymatroid structure, optimal resource allocation and throughput capacities. *IEEE Transactions on Information Theory*, 44(7):2796–2815, November 1998.
60. J. Liu, M. Adler, D. Towsley, and C. Zhang. On optimal communication cost for gathering correlated data through wireless sensor networks. In: *ACM MobiCom*, Los Angeles, CA, USA, 2006.
61. J. Kleinberg and E. Tardos. *Algorithm Design*. Addison Wesley, New York, NY, 2005.
62. S.-Y. Li, R. W. Yeung, and N. Cai. Linear Network Coding. *IEEE Transactions on Information Theory*, 49(2):371–381, 2003.
63. R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11(5):782–795, 2003.
64. T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.
65. P. A. Chou, Y. Wu, and K. Jain. Practical network coding. In: *41st Allerton Conference on Communication, Control, and Computing*, 2003.
66. T. S. Han. Multicasting of correlated multisource to multisink over a network. *preprint*, 2009. [Online]. Available: http://arxiv.org/abs/0901.0608
67. L. Song and R. W.Yeung. Network information flow – multiple sources. In: *Proceedings of Intl Symp. Information Theory*, Washington DC, USA, 2001.
68. A. Ramamoorthy, K. Jain, P. A. Chou, and M. Effros. Separating Distributed Source Coding from Network Coding, *IEEE Transactions on Information Theory*, 52:2785–2795, June 2006.
69. Y. Wu, V. Stankovic, Z. Xiong, and S.-Y. Kung. On practical design for joint distributed source and network coding. *IEEE Transactions on Information Theory*, 55(4):1709–1720, 2009.
70. A. Ramamoorthy. Communicating the sum of sources over a network. In *Proceedings of International Symposium Information Theory*, pages 1646–1650, Toronto, ON, Canada, 2008.
71. A. Ramamoorthy and M. Langberg. Communicating the sum of sources in a 3-sources/3-terminals network. In *Proceedings of International Symposium Information Theory*, pages 2121–2125, Seoul, Korea, 2009.
72. A. Ramamoorthy and M. Langberg. Communicating the sum of sources over a network. *preprint*, 2010. [Online]. Available: http://arxiv.org/abs/1001.5319
73. A. Ramamoorthy. Minimum cost distributed source coding over a network. In: *IEEE International Symposium on Information Theory*, pages 1761–1765, Nice, France, 2007.
74. A. Ramamoorthy, Minimum cost distributed source coding over a network. In: *IEEE Transactions on Information Theory*
75. D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Nashua, NH, 1999.
76. H. D. Sherali and G. Choi. Recovery of primal solutions when using subgradient optimization methods to solve lagrangian duals of linear programs. *Operations Research Letters*, 19(3):105–113, 1996.

Part III
Localization, Time Synchronization, Coordination

Chapter 8
The Spatial Smoothing Method of Clock Synchronization in Wireless Networks

Arvind Giridhar and P.R. Kumar

Abstract Wireless sensor networks are networks of devices which collaborate to perform distributed sensing, processing, and possibly even actuation tasks. In this chapter we consider the problem of the synchronizing clocks in wireless sensor networks. We analyze an approach to clock synchronization, called *spatial smoothing*, that generally synchronizes clocks in a network more accurately than alternative tree-based methods. This approach leads to a distributed least-squares vector estimation problem whose goal is to smooth out the noisy estimates of clock differences of pairs of nodes that can directly exchange packets. We point out connections between the error variance of such a least squares-based clock synchronization and resistance in electrical networks. We determine the limiting clock synchronization accuracy for several types of networks of interest and quantify the improvement over the tree-based method. For random connected wireless sensor networks we show that the clock synchronization error can remain bounded even as the number of nodes in the network increases. This lends support for the feasibility of time-based computation in large networks. We further analyze the convergence time of a distributed iterative algorithm to compute the optimally spatial smoothed estimates. We also propose ways of exploiting the network connectivity graph structure in order to speed up computation.

8.1 Introduction

Knowledge of time can be important in several wireless sensor networks.

Some wireless communication protocols are slotted, and nodes need to agree on slot boundaries. An example is slotted ALOHA. More generally, scheduling actions in a wireless network can improve the performance of the network, for example, by avoiding the possibility of packet collisions.

A. Giridhar (✉)
Goldman Sachs, New York, NY, USA
e-mail: arvind.giridhar@gmail.com

Many wireless sensor networks are used for localization of objects. When this is performed by triangulation of time-of-flight, the clocks of the various nodes need to be synchronized. More generally, many applications or tasks require that all the clocks in the network be synchronized. In one application, described in [1], directional sensors are used to record the times at which objects cross certain lines. The only information available is the set of crossing times. From this, it is shown how the locations of the directional sensors themselves as well as the trajectories of the objects can be estimated. In this example, the accuracy of estimating the positions and trajectories depends crucially on the accuracy with which the clocks at the nodes are synchronized.

In many wireless sensor networks, nodes sleep and wake up on time to receive or send packets to other nodes, thereby saving energy. The accuracy with which sleep schedules can be adhered to determines the duty cycles of nodes, and thus the energy consumptions of nodes. An application described in [2] consists of monitoring a habitat in an environment without any infrastructure for a long period of time. Increasing awake time intervals of nodes to allow for synchronization errors in such situations would typically result in higher energy consumption. To minimize such energy wastage, nodal clocks would have to be tightly synchronized. Accurate clock synchronization is therefore critical to achieving long lifetimes for low duty cycle sensor networks.

Last but not least, coordinated action is also important for control systems, featuring sensors and actuators, deployed over wireless networks. Uncoordinated actions can even destabilize safety-critical networked control systems.

More generally as we move from event-based computation to time-cum-event-based computation, it becomes critically important to synchronize clocks accurately.

The goal of clock synchronization is to provide to each node in a network the reference time, as measured at a designated reference node. Since the clocks at the nodes evolve differently, this can be done by estimating the model of a node's clock with respect to the reference node. In particular each node could calculate its own *skew*, which describes the speed of its clock with respect to the reference node's clock, and its *offset*, which describes the difference between its clock's reading at a particular time and that of the reference clock. In general, the goal of a clock synchronization algorithm is to compute these correction factors for each node. Furthermore, the goal is to obtain a distributed algorithm, where each node utilizes information only from its neighbors. It is also desirable to obtain an algorithm that does not require knowledge of global network topology.

One way to do so is by nodes exchanging time-stamped packets with their neighbors. From this, each node attempts to determine the relative offset with respect to its neighbors, and possibly the relative skew. These local estimates of relative offsets are then pieced together to obtain the global offsets of nodes with respect to a designated reference node.

In [3], a tree-based algorithm is used. A tree rooted at a given reference node is constructed. Then the edge offsets are summed along the path from a given node to the reference node, to yield an estimate of the offset of the node with respect to the reference node. A similar approach is pursued in [4].

An alternate approach is to take advantage of the fact that there are available estimates for every edge in the graph, not just the edges belonging to some spanning tree. Exploiting the fact that the true values of clock differences across edges must satisfy Kirchhoff-like laws, i.e., they must sum to zero around every loop in the graph, yields a set of constraints which can be used to smooth the set of noisy edge offset and skew estimates obtained by the packet exchange procedure. This approach is called *spatial smoothing*. This gives rise, under a simple noise model on the edge estimates, to a distributed parameter estimation problem on graphs.

The contribution of this chapter is a theoretical analysis of this approach. It was first introduced in [5], where a connection was established between the optimal achievable error variance and electrical resistances in topologically identical networks. In [6], an algorithm based on the above notion of spatial smoothing was proposed and implemented, demonstrating superior performance to the tree-based method.

This chapter contains several results from [7, 8], among others. We begin by providing an alternate proof of the equality between the error variance of spatial smoothing and resistance distance by using the different formulations involving vector least-squares estimation and the incidence matrix of the graph introduced in [6]. Using this result, we proceed to study the performance of the least-squares optimal estimate in comparison with the previously proposed tree-based solution. We show that this approach provides considerably better performance than the tree-based approach. For example, it scales much better than the tree-based method for a random connected wireless network. In fact we show that its clock offset error remains bounded even as the number of nodes in the network increases. This lends theoretical support for the feasibility of time-based computation in large wireless sensor network deployments.

We further analyze the convergence properties of the distributed synchronization algorithm proposed in [6], which seeks to converge to the least-squares optimal vector of estimates. We prove convergence and also establish bounds on the settling time in terms of the graph parameters.

For a survey of several previous protocols we refer the reader to [9].

8.2 Synchronizing Two Clocks

We begin by considering the case of just two nodes, numbered 0 and 1, each with a clock. We show how the clocks of the two nodes can be synchronized by packet exchanges between the two nodes. Throughout, we will denote the reference time by t.

We will suppose that the clock of each node i is *affine* with respect to the true reference time, i.e., if $c_i(t)$ is the time reading on clock i when the reference time is t, then

$$c_i(t) = \alpha_i t + v_i \tag{8.1}$$

The quantity α_i will be called the *skew of clock* i, and v_i will be called the *offset of clock* i *at reference time* 0.

We will call

$$o_{ij}(t) := c_j(t) - c_i(t) \tag{8.2}$$

as the *offset of clock* j *from clock* i *at reference time* t. We will also call

$$o_{ij} := v_j - v_i \tag{8.3}$$

as the *offset of clock* j *from clock* i *at reference time* 0. Similarly, we will call

$$\alpha_{ij}(t) := \frac{\alpha_j}{\alpha_i} \tag{8.4}$$

as the *skew of clock* j *with respect to clock* i.

For simplicity, let us denote one of the clocks, say the clock of node 0, as the *reference clock*. Hence its display time is the reference time, i.e., $\boldsymbol{\alpha}_0 = 1$ and $v_0 = 0$.

In order to synchronize the clock of node 1 with respect to the clock of node 0, one needs to determine the values of α_{01} and o_{01}. For this purpose the two nodes exchange time-stamped packets. Consider the ith packet sent from node 0 to node 1. Suppose that clock 0 sends this ith packet at its local time s_{01}^i to node 1. It inserts inside the packet the information about the time s_{01}^i according to its own clock at which this packet is being sent. (This is not strictly necessary. It could communicate this information in a future packet that it sends to node 1). Let us suppose that node 1 receives the packet at time r_{01}^i according to its own clock. Similarly, when node 1 sends its jth packet to node 0, it time-stamps the time s_{10}^j according to its own clock at which it is sending the packet. Node 0 receives the packet at time r_{10}^j according to its own clock. In this way nodes 0 and 1 acquire information about the times they have sent packets and the times at which the other node has received the packets, which they can share with each other in future packets.

We shall suppose that when node 0 sends a packet to node 1, there is a *delay* d_{01}, measured in reference time units, that it takes the packet to reach node 1. By "delay" we do not just mean electromagnetic propagation delay. By delay we mean all the time elapsed from the moment that node 0 time-stamps its packet till the time that node 1 reads the time that it has received the packet. At the transmitter node, this includes the time for the operating system at node 0 to process the packet after time stamping it, followed by the time for the packet to make its way through the rest of the communication protocol stack. Then there is electromagnetic propagation delay. Finally, at the receiver end, there is the time taken by the packet to travel through the communication protocol stack and then the time taken for the clock to be consulted on what time it is. The component of the delay at the transmitter and receiver dominates the electromagnetic propagation delay when the nodes are not very far apart.

Similarly, when node 1 sends a packet to node 0, we suppose that there is a delay d_{10}, again measured in reference time units, for the packet to reach node 0.

To begin with, let us suppose that both these delays are deterministic. That is they do not fluctuate at all from packet to packet.

It should be noted that there are four unknown parameters $(\alpha_{01}, o_{01}, d_{01}, d_{10})$. Nodes 0 and 1 wish to determine these four parameters by exchanging packets with each other.

The relationship between the sending time of the ith packet at node 0 and its receiving time at node 1 is

$$r_{01}^{i} = \alpha_{01}\left(s_{01}^{i} + d_{01}\right) + o_{01} \tag{8.5}$$

(To see the above, note that s_{01}^{i} is the time at which the packet is sent by node 0, measured according to node 0's clock. Subsequently, $\left(s_{01}^{i} + d_{01}\right)$ is the time according to node 0's clock that the packet is received at node i. Converted to the clock of node i this gives the right-hand side of (8.5)). Similarly, when node 1 sends the jth packet to node 0, then the relationship between the time at which it is sent by node 1, and the time at which it is received by node 0, is

$$s_{10}^{j} = \alpha_{01}\left(r_{10}^{j} - d_{10}\right) + o_{01} \tag{8.6}$$

Nodes 0 and 1 obtain the values of $\left(r_{01}^{i}, s_{01}^{i}\right)$ for $i = 1, 2, 3, \ldots$, and $\left(r_{10}^{j}, s_{10}^{j}\right)$ for $j = 1, 2, 3, \ldots$. From this data they would like to determine the values of $\alpha_{01}, o_{01}, d_{01}$, and d_{10}.

This problem can be written as a problem of solving linear equations in many unknowns. Simply define

$$\beta_{01} := \alpha_{01} d_{01} \tag{8.7}$$

$$\beta_{10} := \alpha_{01} d_{10} \tag{8.8}$$

Then (8.5) and (8.6) are linear in $(\alpha_{01}, o_{01}, \beta_{01}, \beta_{10})$:

$$r_{01}^{i} = \alpha_{01} s_{01}^{i} + \beta_{01} + o_{01} \quad \text{for } i = 1, 2, 3, \ldots \tag{8.9}$$

$$s_{10}^{j} = \alpha_{01} r_{10}^{j} - \beta_{10} + o_{01} \quad \text{for } j = 1, 2, 3, \ldots \tag{8.10}$$

However, it is easy to see [10] from the above that one can determine $(\alpha_{01}, \beta_{01} + o_{01}, -\beta_{10} + o_{01})$, but one cannot separately determine $(\alpha_{01}, \beta_{01}, -\beta_{10}, o_{01})$. Or to put it another way, these equations have rank 3, and not 4. Hence four parameters cannot be uniquely determined. (It is of interest, however, that in spite of this a node can predict when the other node will receive a packet, see [11, 12].)

To overcome this problem, let us make one additional assumption:

$$d_{01} = d_{10} \tag{8.11}$$

i.e., the delays in the two directions are *symmetric*. Then these equations can be solved. In fact from just two packets $i = 1, 2$ sent from node 0 to node 1, one can form an estimate $\hat{\alpha}_{01}$ of the skew as

$$\hat{\alpha}_{01} = \frac{r_{01}^2 - r_{01}^1}{s_{01}^2 - s_{01}^1} \tag{8.12}$$

Moreover, from one packet $i = 1$ from node 0 to node 1, and a later packet $j = 1$ from node 1 to node 0, an estimate $\hat{d}_{01}$ can be made as

$$\hat{d}_{01} = \frac{1}{2}\left[\left(r_{10}^1 - s_{01}^1\right) - \left(\frac{s_{10}^1 - r_{01}^1}{\hat{\alpha}_{01}}\right)\right] \tag{8.13}$$

Next, an estimate $\hat{o}_{01}\left(s_{01}^1 + d_{01}\right)$ of $o_{01}\left(s_{01}^1 + d_{01}\right)$ can be made as

$$\hat{o}_{01}\left(s_{01}^1 + d_{01}\right) = r_{01}^1 - s_{01}^1 - \hat{d}_{01} \tag{8.14}$$

From this, the estimate of the offset of node 1 from node 0 at time 0 can be made by

$$\hat{o}_{01}(0) = \hat{o}_{01} = \hat{o}_{01}\left(s_{01}^1 + d_{01}\right) - (\hat{\alpha}_{01} - 1)\left(s_{01}^1 + \hat{d}_{01}\right) \tag{8.15}$$

It is easy to check that in the above deterministic model, the above estimates are all correct, i.e., $\hat{o}_{01} = o_{01}$, $\hat{\alpha}_{01} = \alpha_{01}$, and $\hat{d}_{01} = d_{01}$.

In practice, however, the delays experienced by packets will *not* be deterministic and are better modeled as *random*. The only deterministic component of the delay is the electromagnetic propagation delay. However, when nodes are nearby, this is dominated by the time taken by the operating system to process the packet and transmit it, after the packet has been time-stamped. At the receiving end, there is also the time to process the packet before consulting the clock to record the reception time. There can also be errors in measuring the exact times involved. These quantities are better modeled as random variables. Since the overall delays are therefore also random, the estimates $\hat{\alpha}_{01}$ and $\hat{o}_{01}$ will be erroneous. In Sect. 8.3, when we consider the case of interest in this chapter, which is a network of clocks, our starting point will be the availability of *noisy estimates* of skew and offset for each link. We will suppose that

$$\hat{o}_{01} = o_{01} + N_{01} \tag{8.16}$$

where N_{01} is the error which will be modeled as a random variable.

8.3 A Network of Clocks

Now we turn to the problem of interest, which is a wireless sensor network where each node has a clock. We consider a network consisting of $(n + 1)$ nodes $0, 1, 2, \ldots, n$. We will designate node 0 as the *reference node*, possessing the reference clock. If two nodes can transmit packets to each other, then we connect them by an edge. We will give each such edge an arbitrary orientation. For an oriented edge (i, j), we will call i the head and j the tail. We will also say that the nodes i and j are *neighbors* when they are connected by such an edge. In this way we obtain a directed graph. We will suppose that the graph is connected, i.e., there is a multi-hop undirected path between any two nodes.

In the sequel we will use the *incidence matrix* A of the directed graph. Its rows correspond to the $(n + 1)$ nodes and its columns correspond to the m edges:

$$A_{ij} = \begin{cases} 1 & \text{if node } i \text{ is the head of edge } j \\ -1 & \text{if node } i \text{ is the tail of edge } j \\ 0 & \text{otherwise} \end{cases} \tag{8.17}$$

Since the graph is connected the rank of A is n. Since it is deficit from full rank by 1, we will delete the row corresponding to the root node to obtain a *reduced incidence matrix* which is of full rank. To save on notation, we will continue to denote this reduced incidence matrix by A.

Node 0 is the *reference node*, by which we mean that its clock is the reference clock, whose time will be denoted by t. Each node i has a clock whose display at reference time t is denoted by $c_i(t)$. We will begin our analysis by supposing that the skew of all clocks is 1, i.e., the model of each clock is simply

$$c_i(t) := t + v_i \text{ for } i = 1, 2, \ldots, n \tag{8.18}$$

Then the only difference between any clock i and the reference clock 0 is its offset v_i. The goal of clock synchronization is to determine the *offset* v_i of each clock i from the reference clock at node 0.

Later on, in Sect. 8.6, we will see how the analysis can be extended to the case where there is indeed a skew at each node that is also unknown, i.e.,

$$c_i(t) = \alpha_i t + v_i \tag{8.19}$$

More details can also be found in [6]. The issues arising from the case of randomly varying skew are further addressed in [13]. We focus for the time being on the case of this model (8.18) with only the one unknown parameter of offset, for simplicity of presentation.

8.4 Estimating Node Offsets from Edge Offsets

We will consider a two-step procedure. In the first step the clock offsets between neighboring nodes are estimated. This is done by the neighboring nodes exchanging time-stamped packets between themselves, as described in Sect. 8.2. In the second step these estimates of offsets along edges are combined to estimate the offset of a node's clock with respect to the reference clock.

We begin by revisiting the first step of estimating the offset between neighboring nodes. In Sect. 8.2 we have modeled the delay as deterministic. Then the offset estimate $\hat{o}_{ij}$ is exactly equal to the true offset o_{ij}. However, in practice, the delays experienced by packets will not be deterministic, and are better modeled as random. Hence the estimate $\hat{o}_{ij}$ will be erroneous too. We will suppose that for each edge (i, j) in the network, the estimate $\hat{o}_{ij}$ is related to the true value o_{ij} by

$$\hat{o}_{ij} = o_{ij} + N_{ij} \tag{8.20}$$

where N_{ij} is a random variable that captures the sum of all the random effects. We model it as a normal random variable with mean 0 and variance σ^2. (The distribution is not specifically important to our analysis; however, the fact that the noise variables are independent and mean zero is important.) We will assume that the random errors $\{N_{ij} : (i, j) \text{ is an edge}\}$ are independent and identically distributed (i.i.d.).

The next issue is how to piece together these estimates $\hat{o}_{ij}$ of the offsets between neighboring clocks to obtain an estimate of the offset of a node from the reference clock. The most straightforward way to do so is to choose a path connecting a node i with the reference node 0 and then to simply add up the offsets along the edge. That is, if $\{(i_0, i_1), (i_1, i_2), \ldots, (i_{k-1}, i_k)\}$ is a path connecting $0 = i_0$ with $i_k = i$ through a series of edges $(i_{\ell-1}, i_\ell)$, then the estimate of the offset of node i from the reference node 0 is

$$\hat{o}_{i0} = \sum_{\ell=1}^{k} \hat{o}_{i_{\ell-1}, i_\ell} \tag{8.21}$$

In a tree, there is only one path, but in a general graph there may be more than one path and so one would need to choose one of the paths.

How good is this estimate? To determine this, note that the sum of k independent $N(0, 1)$ random variables is $N(0, k)$. That is, the variance increases linearly in the number of variables being summed, i.e., like k. So the standard deviation increases like $\sqrt{k}$, the square root of the number of variables being added up. Since the number of edges between the reference node and any node is bounded by the diameter of the graph, i.e., the maximum number of edges needed to connect any node with any other node, we see that

$$\tilde{o}_{i0} = O\left(\sqrt{\text{Diameter of graph}}\right) \tag{8.22}$$

where $\tilde{o}_{i0} := \hat{o}_{i0} - o_{i0}$ is the error in the estimate.

Let us begin by examining how accurate this is.

Example: A tree. Note that if the graph of the network is a *tree*, then since the diameter of a tree is $O(n)$, we see that the error is $O\left(\sqrt{n}\right)$.

Other graphs can have diameters less than $O(n)$; we consider next the class of *geometric graphs*.

8.4.1 Geometric Graphs

We now give the wireless sensor network a more physical character by supposing that the $(n+1)$ nodes $0, 1, 2, \ldots, n$ are located on a plane. Denote the distance between two nodes i and j by ρ_{ij}. We will suppose that node i can send a packet to node j if $\rho_{ij} \leq r$. Here r is the *range* of communication of the radios at nodes. We will suppose, as before, that node 0 is the *reference node*, by which we mean that its clock is the reference clock.

To examine how good the estimates generated by (8.22) are, we consider several cases.

Example: A collocated network. Suppose that the range r is so large that $r \geq \rho_{ij}$ for all i, j. Then all nodes can communicate directly with each other. We call such a network a *collocated network*, see Fig. 8.1. The resultant graph is a clique on n vertices. Its diameter is 1. Hence the synchronization error is $\Theta(1)$. While the error variance is bounded, it does not go to 0 as the number of nodes $n \to \infty$, which is however what one would desire in such a collocated scenario.

Example: A grid or integer lattice network. Suppose that the $(n+1)$ nodes are located at positions with x-coordinates and y-coordinates that are integers, i.e., at positions (ℓ, k) where $1 \leq \ell, k \leq \sqrt{n+1}$ are integers. We call this a *grid* or *integer lattice* network, see Fig. 8.2. Suppose also that the range $r = 1$. Then every node can communicate with its nearest neighbors. The diameter of the graph is $\Theta\left(\sqrt{n}\right)$. Therefore the synchronization error is $\Theta(n^{1/4})$. Hence we see that the synchronization error grows polynomially in n.

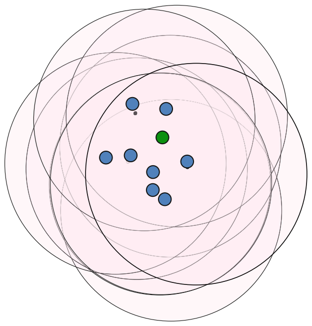


Fig. 8.1 A collocated network

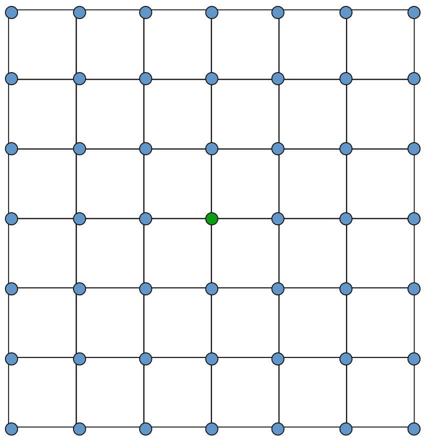


Fig. 8.2 A grid or integer lattice network

Example: A random network that is connected. Consider now a unit area square in the plane. Suppose that nodes are randomly distributed in the square, with probability distribution that is uniform, and independently of each other. This can model the deployment of the nodes of a wireless sensor network. The first issue in such a wireless network is that one would like to choose the range $r(n)$, possibly as a function of the number of nodes, in such a way that the graph of the network is *connected*, see Fig. 8.3. Clearly, as n increases there are more nodes packed into the unit area square, and hence a smaller value of range should suffice. That is, if the range $r(n)$ chosen can depend on n, then one expects that $r(n)$ can decrease with n. At what precise rate can it decrease and still be connected? This is answered in [14, 24].

Theorem: Connectivity of a geometric random graph *Suppose that the range of nodes is $r(n)$, where, as denoted, the range can depend on the number of nodes. Consider $r(n) = \sqrt{\frac{\log n + \psi_n}{n}}$. Then*

$$\lim_{n \to +\infty} \text{Prob}(\textit{Graph is connected}) = 1 \Longleftrightarrow \lim_{n \to +\infty} \psi_n = +\infty \qquad (8.23)$$

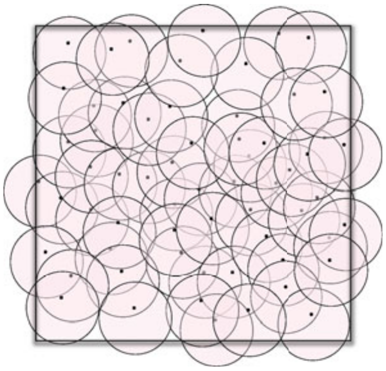


Fig. 8.3 A random connected network

In particular, note that if $r(n) = \sqrt{\frac{2 \log n}{n}}$, then the graph of the network is connected with high probability (whp) converging to 1 as $n \to \infty$.

Let us suppose that the range has been chosen so that the graph is indeed connected. We shall call such a graph as a *connected geometric random graph*. Then since the maximum distance between any two points in the unit area square is $\sqrt{2}$, it follows that the maximum number of hops to connect any two nodes is $O\left(\sqrt{\frac{n}{\log n}}\right)$ in any graph that is connected whp. From this it follows that

$$\text{Error} = O\left(\left(\frac{n}{\log n}\right)^{1/4}\right) \tag{8.24}$$

Hence, again the error can grow polynomially in the number of nodes.

In all these cases, one hopes that one can do better. In the collocated network, one would like to have an error that is $o(1)$, i.e., Error $\to 0$ as the number of nodes $n \to +\infty$. In the connected geometric random graph, one would like Error = $O(1)$, i.e., the error to be bounded as the number of nodes increases. If the latter is feasible, then one will be able to achieve bounded error of clock synchronization even in large wireless sensor networks. This will lend theoretical support for the feasibility of large sensor networks where accurate time-stamping is important for inference or control.

To achieve the above goals we will exploit the *network* aspect of the system as well as the physical property of *time*.

8.5 Spatial Smoothing

The key property that we will exploit to improve the accuracy of clock synchronization is the nature of *time*. Consider any *cycle* or *loop* in the graph, i.e., any sequence of edges, $L = \{(i_0, i_1), (i_1, i_2), i_2, i_3), \ldots, (i_{\ell-1}, i_\ell = i_0)\}$. Then the sum of the *true* edge clock differences added up along the edges in the loop must satisfy

$$\sum_{(i,j) \in L} o_{ij} = 0, \text{ for every loop } L \tag{8.25}$$

This is analogous to Kirchhoff's voltage law. The role of node voltages is played by the nodal offsets $\{v_i\}$, and the role of potential difference across an edge is played by the edge offset o_{ij}.

Observe that a graph may in general have several cycles. For each of these cycles property (8.25) holds.

Let us denote the n-dimensional vector of node offsets by $v = (v_1, v_2, \ldots, v_n)^T$, and the corresponding m-dimensional vector of edge offsets by o. Then v and o are related by

$$o = A^T v \tag{8.26}$$

Note that the orientation of the graph is chosen according to the signs of the edge offsets. From the above it follows that the true offset vector o must belong to the range space of the transpose of the reduced incidence matrix A. Note that any vector o of edge offset values arising from a vector v of nodal values through the above relation (8.26) will necessarily satisfy (8.25).

While the above property (8.25) is satisfied by the *true* edge offsets, it will generally *not* be satisfied by the *noisy estimates* of the edge offsets. Note now that our goal is to obtain estimates of the *node offsets* of the *nodes* in the graph. Thus, we can formulate the synchronization problem as one of estimating the vector v of node clock offsets, given the vector of noisy edge estimates $\hat{o} = A^T v + N$. We can pose this as a least-squares optimization problem.

We now describe such a least-squares synchronization approach for estimating the clock offsets of all nodes from the reference clock [6]. In effect we have a non-Bayesian vector parameter estimation problem. The noise vector N is i.i.d. Gaussian, and consequently the maximum likelihood estimate is given by a *least-squares fit* of the offset estimate vector to the range space of A^T, i.e., the space of offset vectors satisfying the network Kirchhoff constraints. The least-squares optimal vector estimate is the solution to the following quadratic minimization problem:

$$\hat{v} = \arg\min_{v} \parallel A^T v - \hat{o} \parallel^2 \tag{8.27}$$

The above formulation of the estimation problem as a standard quadratic optimization problem is taken from [6] and suggests a distributed method to compute the least-squares optimal estimate, as we will subsequently show in Sect. 8.8. An alternate method to obtain the same optimal estimate was proposed in [5].

In effect we have obtained a two-step procedure. First, for each pair of nodes (i, j) that are neighbors, through bilateral packet exchanges between neighboring nodes, as explained in Sect. 8.2, we obtain an estimate $\hat{o}_{ij}$ of the offset between the two neighboring nodes i and j. This estimate *only* takes into account the information contained in the time-stamps of the packets sent from node i to node j or vice-versa. However, it does *not* in any way take into account any information contained in the time-stamps of packets exchanged between any two *other* nodes (k, ℓ).

On the other hand, one does want to take into account all information that is available, in arriving at estimates $(\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_n)$ of the offsets of the nodes. Thus we need to stitch all the bilateral estimates $\{\hat{o}_{ij} : i \text{ and } j \text{ are neighbors}\}$ to produce estimates of the nodal offsets $\{v_i : i \text{ is a node}\}$. We do this by a least-squares fit (8.27).

We call the second step of this procedure as *spatial smoothing*. It smooths out the noise in each individual estimate $\hat{o}_{ij}$ by relying on all the other estimates $\{\hat{o}_{kl}\}$ made all over the network.

There are two questions that arise:

(i) How does one perform such spatial smoothing in a distributed fashion to obtain node offsets from edge offsets?

(ii) How well does spatial smoothing approach perform? What is its accuracy, and how fast does it converge?

We address these questions in the sequel.

8.6 Estimating Nodal Skews

At this point we digress briefly to discuss how clocks with general skews can be handled. If we consider the more general model (8.19) that allows for different skews for different clocks, then it can be handled in a similar manner to the way offsets are handled. First note that if $\{(i_0, i_1), (i_1, i_2), \ldots, (i_{k-1}, i_k)\}$ is a path connecting $0 = i_0$ with $i_k = i$ through a series of edges $(i_{\ell-1}, i_\ell)$, then

$$\alpha_{i0} = \prod_{\ell=1}^{k} \alpha_{i_{\ell-1}, i_\ell} \tag{8.28}$$

This is a multiplicative analog of the additive relation (8.21) that holds for offset estimates. Hence, instead of dealing with $\boldsymbol{\alpha}_{ij}$, if we deal with its natural logarithm $\log(\alpha_{ij})$, we obtain

$$\log(\alpha_{i0}) = \sum_{\ell=1}^{k} \log(\alpha_{i_{\ell-1}, i_\ell}) \tag{8.29}$$

which is the exact analog of (8.21). It shows that methods for estimating offsets have multiplicative counterparts that can be used to estimate skews.

Using this, spatial smoothing can be extended to the case of estimating skew. Instead of (8.25), one has

$$\sum_{(i,j)\in L} \log(\alpha_{ij}) = 0 \tag{8.30}$$

for every loop L. Thus one obtains a relation that is analogous to (8.25), following which one obtains an optimization problem that is analogous to (8.27). This results in a two-step procedure for piecing together relative skew estimates (8.12) that are obtained by bilateral packet exchanges between nodes, to produce skew estimates of nodes with respect to the reference clock.

8.7 Properties of the Least-Squares Solution

We return to the problem of estimating offsets for the one-parameter model (8.18) by the least-squares estimate (8.27). First we note some nice properties of the least-squares estimate $\hat{v}$ and its error

$$\tilde{v} := \hat{v} - v \tag{8.31}$$

Theorem 1 *1. The optimal solution of (8.27) is*

$$\hat{v} = (AA^T)^{-1}A\hat{o} \tag{8.32}$$

It is unique.

2. The least-squares estimate is unbiased, i.e., $E(\hat{v}) = v$. Its error covariance is $E[\tilde{v}\tilde{v}^T] = \sigma^2(AA^T)^{-1}$. The least-squares optimal solution $\hat{v}$ is a minimum variance unbiased estimate and achieves the Cramer–Rao lower bound.

3. $\hat{v}$ is the unique minimum variance unbiased estimate.

Proof (i). Since the matrix AA^T is positive definite, the unique minimizing solution is obtained by setting the gradient to 0.

(ii). Let v be the true vector of node offsets and N the noise vector. Because the noise N_{ij} is zero mean, we have

$$E[\hat{v}] = E[(AA^T)^{-1}A\hat{o}] = (AA^T)^{-1}AA^Tv + E[N] = v \tag{8.33}$$

Thus, the estimate is unbiased. Now, consider the $n \times n$ Fisher information matrix I_v with ijth element given by

$$(I_v)_{ij} = E\left[\left(\frac{\partial}{\partial v_i}\log p_v(\hat{o})\right)\left(\frac{\partial}{\partial v_j}\log p_v(\hat{o})\right)\right] \tag{8.34}$$

where $p_v(\hat{o})$ is the probability density function of the vector $\hat{o}$ given node offset vector v. We have

$$\begin{aligned}\frac{\partial}{\partial v_i}\log p_v(\hat{o}) &= \frac{\partial}{\partial v_i}\left(\frac{1}{2\sigma^2}\sum_k\left(\hat{o}_k - A_k^Tv\right)^2\right) + \text{constant}\\ &= \frac{1}{\sigma^2}\sum_k\left(\hat{o}_k - A_k^Tv\right)(A^T)_{ki}\end{aligned} \tag{8.35}$$

Therefore, we have

$$\begin{aligned}(I_v)_{ij} &= E\left[\frac{1}{\sigma^4}\sum_k\sum_l\left(\hat{o}_k - A_k^Tv\right)\left(\hat{o}_l - A_l^Tv\right)A_{ki}^TA_{lj}^T\right]\\ &= E\left[\frac{\sigma^4}{\sum_k}\sum_l N_kN_lA_{ki}^TA_{lj}^T\right]\\ &= \frac{1}{\sigma^4}\sigma^2\sum_k A_{ik}A_{kj}^T\\ &= \frac{1}{\sigma^2}(AA^T)_{ij}\end{aligned} \tag{8.36}$$

The Cramer–Rao bound [15] states that for any unbiased estimate $\hat{v}$, the error covariance matrix satisfies $E[(\hat{v}-v)(\hat{v}-v)^T] \geq I_v^{-1}$. We now evaluate the error covariance matrix of the least-squares optimal solution $\hat{v}$. The error vector is given by

$$\begin{aligned}\tilde{v} &= \hat{v} - v \\ &= (AA^T)^{-1}A(A^Tv + N) - v \\ &= (AA^T)^{-1}AN \end{aligned} \tag{8.37}$$

The error covariance is therefore $E[\tilde{v}\tilde{v}^T] = \sigma^2(AA^T)^{-1}$. Thus, $E[\tilde{v}\tilde{v}^T] = (I_v)^{-1} = \sigma^2(AA^T)^{-1}$. Thus, the error covariance matrix of $\hat{v}$ achieves the Cramer–Rao bound and is therefore the minimum variance unbiased vector estimate of v.

(iii). Consider the problem of computing the minimum variance linear unbiased estimate of the offset (with respect to the root node) of a given node i, given the edge offset estimate vector $\hat{o}$. That is, we wish to find a vector h such that

$$\begin{gathered}\min E[v_i - h^T\hat{o}]^2 \\ \text{subject to} \\ E[h^T\hat{o}] = v_i\end{gathered} \tag{8.38}$$

From part (i), we know that $h^* = (A^T(AA^T)^{-1})_i$ (the subscript i denotes the ith column of the matrix) is a solution to this problem. We claim that this is the unique solution. If so, the same is true for any choice of a node i, which proves the result.

Since $\hat{o} = A^Tv + N$ and N is a zero mean vector, the constraint becomes $h^TA^Tv = v_i$. This has to be true for an arbitrary vector v, which implies that $Ah = e_i$, where e_i is the n-dimensional vector with ith entry 1 and all other entries 0. Modifying the objective function (8.38), we have

$$\begin{aligned}E[v_i - h^T\hat{o}]^2 &= E[h^TN]^2 \\ &= \sigma^2h^Th\end{aligned} \tag{8.39}$$

Thus, we have the equivalent optimization problem

$$\begin{gathered}\min h^Th \\ \text{subject to} \\ Ah = e_i\end{gathered} \tag{8.40}$$

The Lagrangian of this problem is $h^Th + \lambda^T(Ah - e_i)$, where λ is the vector of Lagrange multipliers. Minimizing this with respect to h yields the solution $h^* = \frac{1}{2}A^T\lambda$. Maximizing the dual objective function $\frac{1}{4}\lambda^TAA^T\lambda - \lambda^T\left(\frac{1}{2}AA^T\lambda - e_i\right)$ over all values of $\boldsymbol{\lambda}$ gives the solution to the primal optimization problem. This solution is easily obtained by taking the gradient with respect to $\boldsymbol{\lambda}$, giving the unique solution

$$\lambda^* = 2(AA^T)^{-1}e_i$$

Thus, the solution to the original problem is also unique and given by $h* = \frac{1}{2}A^T\lambda^*$, which gives us back the original solution to the least-squares problem. ■

Note that the proof of part (iii) also provides an alternate proof that the least-squares optimal solution is the same as the minimum variance unbiased estimate.

The following result, previously established in [5], shows an interesting connection between the performance of the least-squares optimal solution and resistances in electrical networks. We provide an alternate proof below [7].

Theorem 2 *(Karp et al. [5]) Replace each edge in the graph by a 1 Ω resistor. Then the error variance of node i is*

$$E[\hat{v}_i - v_i]^2 = \sigma^2 \times (\textit{Electrical resistance between nodes } i \textit{ and } 0) \tag{8.41}$$

Proof From electrical network theory [16], it is known that the ith diagonal entry of $(AA^T)^{-1}$ is the resistance between nodes i and 0. This can be obtained by computing the voltage resulting from the addition of current sources between every node i and the root. The theorem follows immediately by noting that $E[\tilde{v}\tilde{v}^T] = \sigma^2(AA^T)^{-1}$, and its ith diagonal entry is the variance of $\tilde{v}_i$. ■

The connection to electrical resistance via the Laplacian matrix is also exploited in [17], which considers the problem of node localization in wireless networks.

Let us call the resistance between nodes i and 0 in the above electrical network as the *resistance distance between the two nodes*. From the above theorem it follows that to evaluate the performance of the spatial smoothing algorithm, all we need to do is evaluate or at least bound the resistance distances. Moreover, from the electrical analog, several properties become more intuitive:

- Adding more bilateral estimates between nodes is tantamount to adding more edges. Hence it decreases the electrical resistances and thus improves the accuracy of clock synchronization.
- From the above it follows that adding edges to a tree improves the accuracy of clock synchronization. Hence spatial smoothing performs better than the earlier described tree-based estimates.
- Adding parallel paths decreases the conductance, which is the reciprocal of resistance. Hence taking advantage of estimates over parallel paths improves the accuracy of clock synchronization by averaging the effect of noise.
- For a tree the resistance distance is just the sum of the variances over the edges in the unique path. Hence the variance of the synchronization error increases linearly with the number of edges along the path from a node to the reference node. The tree-based algorithm [3] is optimal in the least-squares sense in this case, achieving the above error variance.

We now revisit the canonical graphs considered earlier to assess the benefits of spatial smoothing by the least-squares method [7]. We do this by evaluating the maximum error variance $R_{\max}$ (equivalently, maximum resistance) of the estimated offset of any node, for the graphs that were considered earlier.

Example: A tree. $R_{\max} = O(\text{diameter})$, as noted earlier.

Example: A collocated network. For a clique, $R_{\max} = 2/n$. This is easily derived using Kirchhoff's laws and symmetry. Hence, the error of the least-squares solution converges to 0 as the number of nodes increases. In contrast, if one does not employ spatial smoothing and just relies on the bilateral estimates only between a node i and the reference node, then the error is $\Theta(1)$. Spatial smoothing results in the much better accuracy of $\Theta\left(\sqrt{1/n}\right)$.

Example: A grid or integer lattice network. Determining the resistance of an infinite lattice is a classical problem in electrical circuit theory. An expression for the resistance between points in a finite lattice has been derived in [18]. We show below that the resistance between opposite corners of a square lattice of n nodes is $\Theta(\log n)$. This shows that spatial smoothing has a much better accuracy of $\Theta\left(\sqrt{\log n}\right)$ than the tree-based scheme in large networks since the error in the latter case grows like $\Theta\left(\sqrt{n}\right)$.

Theorem 3 *The maximum resistance between any two nodes in the square lattice of n nodes is* $\Theta(\log n)$.

Proof The problem of computing resistances between points on a grid goes back to the classical physics problem of computing resistances between adjacent points on an infinite grid.

Closed form expressions for the resistance between points on different types of finite lattice resistive networks have been derived in [18]. These formulae are expressed in terms of the eigenvalues and eigenvectors of the Laplacian matrices of the respective networks. In particular, the resistance between two points (x_1, x_2) and (y_1, y_2) in an $M \times N$ grid of resistors, where the "horizontal" resistors have value r and the "vertical" resistors have value s, is shown to be the following:

$$\begin{aligned} R((x_1,x_2),(y_1,y_2)) &= \frac{r}{N}|x_1-x_2| + \frac{s}{M}|y_1-y_2| + \frac{2}{MN} \\ &\times \sum_{m=1}^{M-1}\sum_{n=1}^{N-1} \\ &\frac{\left(\cos\left(x_1+\frac{1}{2}\right)\theta_m \cos\left(y_1+\frac{1}{2}\right)\phi_n - \cos\left(x_2+\frac{1}{2}\right)\theta_m \cos\left(y_2+\frac{1}{2}\right)\phi_n\right)^2}{\dfrac{1-\cos\theta_m}{r}+\dfrac{1-\cos\phi_n}{s}} \end{aligned} \tag{8.42}$$

where $\theta_m = \frac{m\pi}{2M}$ and $\phi_n = \frac{n\pi}{2N}$.

Here, we use this formula to derive an upper bound on the resistance between corner points of an $M \times M$ grid of unit resistors [7]. The lower bound follows similarly, but we omit the proof.

First, we need a couple of simple lemmas.

Lemma 1 *For* $0 < x \leq \pi/2$,

$$\sin x \geq \frac{2}{\pi} x$$

Proof We prove that $\frac{\sin x}{x}$ is a decreasing function of x for $0 < x \leq \pi/2$. The result follows since $\frac{\sin x}{x} < \frac{\sin \pi/2}{\pi/2}$.

$$\begin{aligned} \frac{d\frac{\sin x}{x}}{dx} &= \frac{x \cos x - \sin x}{x^2} \\ &= (x - \tan x)\frac{\cos x}{x^2} \end{aligned}$$

Now $x - \tan x = 0$ at $x = 0$, and $\frac{d(x - \tan x)}{dx} = 1 - \sec^2 x < 0$ for $0 < x \leq pi/2$. Therefore $x - \tan x < 0$ for $0 < x \leq \pi/2$, and so we are done. ∎

Lemma 2 *For* $M \geq 1$,

$$\sum_{m=1}^{M-1} \sum_{n=1}^{N-1} \frac{1}{n^2 + m^2} \leq 1 + \log M \tag{8.43}$$

Proof We make use of the following simple inequality:

$$\sum_{m=L}^{M} \frac{1}{m^2} < \int_{L-1}^{M} \frac{1}{x^2} dx \tag{8.44}$$

Therefore,

$$\begin{aligned} \sum_{m=1}^{M-1} \sum_{n=1}^{N-1} \frac{1}{n^2 + m^2} &< \sum_{m=1}^{M-1} \sum_{n=1}^{N-1} \frac{2}{(n+m)^2} \\ &< \sum_{m=1}^{M-1} \int_{m}^{m+M-1} \frac{1}{x^2} dx \\ &= \sum_{m=1}^{M-1} \frac{1}{m} - \frac{1}{m + M - 1} \\ &< 1 + \int_{1}^{M} \frac{1}{x} dx \\ &= 1 + \log M \end{aligned}$$

For the corner points in an $M \times M$ grid, (8.42) simplifies to

$$R((0,0),(M,M)) = 2 + \frac{1}{M^2}\sum_{m=1}^{M-1}\sum_{n=1}^{N-1} \frac{\left(\cos\left(\frac{m\pi}{2M}\right)\cos\left(\frac{n\pi}{2N}\right) - \cos\left(m\pi + \frac{m\pi}{2M}\right)\cos\left(n\pi + \frac{n\pi}{2M}\right)\right)^2}{\sin^2\frac{m\pi}{2M} + \sin^2\frac{n\pi}{2M}} \tag{8.45}$$

Applying Lemma 1, we have the inequality

$$\begin{aligned} R((0,0),(M,M)) &\le 2 + \sum_{m=1}^{M-1}\sum_{n=1}^{N-1} \frac{\left(\cos\left(\frac{m\pi}{2M}\right)\cos\left(\frac{n\pi}{2N}\right) - \cos\left(m\pi + \frac{m\pi}{2M}\right)\cos\left(n\pi + \frac{n\pi}{2M}\right)\right)^2}{m^2+n^2} \\ &< 2 + 4\sum_{m=1}^{M-1}\sum_{n=1}^{N-1}\frac{1}{n^2+m^2} \\ &< 6 + 4\log M \end{aligned} \tag{8.46}$$

where the last inequality follows from Lemma 2.

Thus, the resistance between corner points in an $M \times M$ grid is upper bounded by $6 + 4\log M$. ■

Example: A random network that is connected The following result gives the order of the resistance in a connected geometric random graph. It shows that the synchronization error is bounded in large random connected planar wireless networks. It thus gives theoretical support to the possibility of time-based computation in large wireless sensor networks.

Theorem 4 *(Giridhar and Kumar [7]) The maximum resistance between any two nodes in the random planar network with common transmission range $c\sqrt{\frac{\log n}{n}}$, $c > 5\log(4/e)$, is $O(1)$.*

Proof We make use of a couple of known properties of the random planar network.

- Recall that if $r(n) = \sqrt{\frac{2\log n}{n}}$, the graph of the network is connected with high probability converging to 1 as $n \to \infty$ [14]. The general statement is that a range of $\Theta\left(\sqrt{\frac{\log n + \psi_n}{n}}\right)$ with $\psi_n \to \infty$ is necessary and sufficient for the network graph to be connected with probability going to one as $n \to \infty$.
- We also need the following lemma, which is proved in [19].

Lemma 3 *Consider the tessellation of the unit square into $m = \left\lceil\sqrt{\frac{n}{K\log n}}\right\rceil^2$ equally sized squares $s_1, s_2, \ldots, s_m$. Let n_i be the number of nodes in square s_i. If $K > 1/\log(4/e)$, there exist constants $\mu_1, \mu_2 > 0$ such that*

$$\lim_{n\to\infty} P\left[\max_i n_i < \mu_1 \log n\right] = 1 \tag{8.47}$$

and

$$\lim_{n\to\infty} P\left[\min_i n_i > \mu_2 \log n\right] = 1 \tag{8.48}$$

In other words, for such a tessellation, the number of nodes in each cell is no more than $c \log n$ with high probability, uniformly over all such cells in the unit square.

As a direct consequence of Lemma 3, we have the following corollary.

Corollary 1 *Let the unit square be tessellated into $m = \left\lceil \sqrt{\frac{n}{K \log n}} \right\rceil^2$ equally sized square cells. Further, consider the network graph obtained by connecting every pair of nodes which are either in the same cell or in adjacent cells by an edge. Then, with probability converging to 1 as n tends to ∞, there are $\mu_2 \log n$ disjoint m node grid subgraphs, with each subgraph having one node per cell.*

Proof As a consequence of Lemma 3, all cells have at least $\mu_2 \log n$ nodes. We can thus construct $\mu_2 \log n$ disjoint subgraphs by picking one node per cell for each subgraph. Each of the resulting subgraphs will be an m node grid due to the adjacency properties of the original network.

We can now bound the resistance between any two nodes in the random planar network. Without loss of generality, consider two nodes which are in diametrically opposite cells in the network. A similar argument holds for any other pair of nodes as well.

Let $K = \log(4/e)$, and divide the unit square into $m = \left\lceil \sqrt{\frac{n}{K \log n}} \right\rceil^2$ square cells. The maximum distance between any two points in adjacent squares is $\sqrt{5}l$, where $l = \sqrt{\frac{n}{K \log n}}$ is the length of an individual square cell, as can be seen in Fig. 8.4.

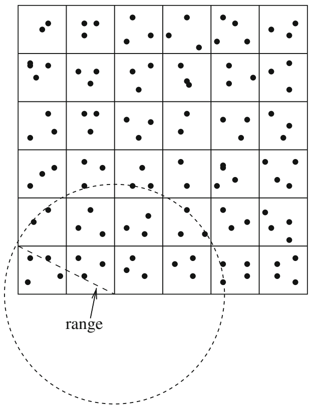


Fig. 8.4 Unit square divided into cells

Setting the transmit range $r(n)$ to be larger than $\sqrt{5K\frac{\log n}{n}}$ ensures that every pair of nodes in adjacent cells will be connected and also that the network as a whole is connected. Furthermore, the conditions of Lemma 3 are also satisfied, implying that with probability going to 1 as $n \to \infty$, every square cell will have at least $\mu_2 \log n$ nodes.

Now, take any two nodes. It is sufficient to consider nodes in the diagonally opposite corner cells of the unit square. Similar reasoning applies for any other pair of nodes. By Corollary 1, there exist $\mu_2 \log n$ disjoint grids, each with nodes in every cell. Consider the electrical network consisting of these $\mu_2 \log n$ grids along with the two nodes in question, along with all the edges from each of these two nodes to the $\mu_2 \log n$ grid nodes in their respective cells. This electrical network is a subnetwork of the original network graph, and so the resistance between the two nodes in this network is a lower bound for the resistance in the original network.

But this network simply consists of $\mu_2 \log n$ parallel disjoint subgraphs linking the two nodes. The analysis of grid resistance above shows that each of these grids has resistance less than $6 + 4\log n$. Thus, the overall resistance is less than $\frac{1}{\mu_2 \log n}(8 + 4\log n)$, which itself is less than $8/\mu_2$. ■

From the above examples of networks we see that the spatial smoothing algorithm gives much greater accuracy than tree-based synchronization.

8.8 The Distributed Spatial Smoothing Algorithm Based on Coordinate Descent

The next question that arises is how to compute the least-squares optimal estimate $\hat{v}$ in a distributed manner. That is we would like to use a *distributed algorithm* where each node only employs information that it obtains from its neighbors. Further, we would like to obtain a procedure that does *not* rely on the knowledge of the *network topology* that knowledge of A entails. Note that in a mobile ad hoc network, keeping track of network topology can be challenging, as can be the problem of further maintaining a tree.

We will employ *coordinate descent* to solve both these problems. Let $F(v) :=\| A^T v - \hat{o} \|^2$. At each step, one node i minimizes the objective function $F(v)$ with respect to just the single variable v_i.

To determine the minimizing v_i, note that

$$\frac{dF(v)}{dv_i} = (AA^T)_i v - A_i \hat{o} \tag{8.49}$$

Exploiting the structure of the reduced incidence matrix A shows that

$$(AA^T)_{ij} = \begin{cases} D_i & \text{if } i = j \\ -1 & \text{if } i \text{ and } j \text{ are neighbors} \\ 0 & \text{otherwise} \end{cases} \tag{8.50}$$

where D_i is the number of neighbors that node v_i has, i.e., its degree. Hence, (8.49) simplifies to

$$\frac{dF(v)}{dv_i} = D_i v_i - \sum_{\{j:j \text{ is a neighbor of } i\}} (v_j + \hat{o}_{ji}) \tag{8.51}$$

The minimum is obtained by setting the above to 0:

$$v_i = \frac{1}{D_i} \sum_{\{j:j \text{ is a neighbor of } i\}} (v_j + \hat{o}_{ji}) \tag{8.52}$$

The resulting algorithm is therefore particularly simple. Each node simply updates its estimate of its offset with respect to the reference node 0 as the average of all its neighbors' estimates plus the offset estimates of the corresponding edges connecting those neighbors to itself. (To put it another way, this is the average of its neighbors' estimates of *its own estimate*.) Thus, nodes need to only communicate with their neighbors. Hence we have a distributed algorithm. Furthermore, the second goal is also realized since no node needs to know the global topology of the network.

We summarize the resulting spatial smoothing algorithm, proposed in [6]:
Distributed Asynchronous Spatial Smoothing:

- Each node regularly exchanges time-stamped packets with each of its neighbors and estimates the corresponding bilateral offsets.
- Each node regularly broadcasts its current estimate of its offset v_i to all its neighbors.
- Each node regularly averages the value of $(v_j + \hat{o}_{ji})$ that it has obtained from its neighbors.

These steps can be combined or run in parallel. They can be performed asynchronously too.

The synchronous version of the above algorithm is the following.
Distributed Synchronous Spatial Smoothing: The bilateral estimates are determined as earlier. However, all the nodal clock offsets are updated synchronously:

$$v_i^{(k+1)} = \frac{1}{D_i} \sum_{\{j:j \text{ is a neighbor of } i\}} \left(v_j^{(k)} + \hat{o}_{ji}\right) \text{ for all nodes } i = 1, 2, \ldots, n \tag{8.53}$$

Here k is the common sequence number of all the updates.

8.9 Convergence Analysis of the Spatial Smoothing Algorithm

In Sect. 8.7 we have determined how accurate the spatial smoothing-based least-squares solution is. In this section we will first show that the algorithm does indeed converge to the least squares solution; see [25]. We will further analyze how long the algorithm takes to converge to a solution that is nearly the final least-squares estimate [7]. Thus Theorem 2 provides the property of the asymptotic solution, while the theorems in this section inform us about the time taken to get close to this asymptotic estimate. This time will also be related to the number of nodes in the network, indicating the scaling performance.

Theorem 5 *Both the synchronous and asynchronous spatial smoothing algorithms converge to the least-squares solution $\hat{v}$.*

Proof The asynchronous coordinate descent approach is well known to converge for quadratic costs, see [20]. So we only consider the synchronous version, which can be written in vector notation as

$$v^{(k+1)} = v^{(k)} - D^{-1}(AA^T v^{(k)} - A\hat{o}) \tag{8.54}$$

Defining $\bar{v}^k := v^{(k)} - (AA^T)^{-1}A\hat{o}$, we only need to show that $\bar{v}^k$ converges to 0. It can be written as

$$\bar{v}^{(k+1)} = M\bar{v}^{(k)} \tag{8.55}$$

where $M := I - D^{-1}AA^T$. We only need to show that the spectral radius of M, denoted $\rho(M)$, is strictly less than 1.

Note that

$$M_{ij} = \begin{cases} 0 & \text{if } i = j \\ 1/D_i & \text{if } i \neq j \end{cases} \tag{8.56}$$

Hence all row sums of M are equal to 1, except for nodes which are neighbors of the reference node. For a neighbor i of the reference node, the row sum is $\frac{D_i - 1}{D_i}$. Thus, matrix M is a nonnegative, substochastic matrix. Further it is irreducible, and at least one row sum is strictly less than 1. Hence $\rho(M) < 1$ (see [21], Problem 8.3.7). ■

Next we obtain [7] a bound on the number of steps, $T_\epsilon(A, v^{(0)})$, taken by the synchronous algorithm started with an initial estimate $v^{(0)}$, to converge to an ε-neighborhood of the final estimate $\hat{v}$. We will call this the *ε-settling time of the algorithm*. It involves the *edge-connectivity* parameter κ of the connected graph, which is defined as the minimum number of edges that, when deleted, disconnect the graph.

Theorem 6 *The settling time is bounded as follows:*

$$\frac{\sum_{1\le i\le n} D_i}{D_0}\log\frac{1}{\varepsilon \parallel v^{(0)} \parallel} < T_\varepsilon(A, v^{(0)}) \tag{8.57}$$

and

$$T_\varepsilon(A, v^{(0)}) < \frac{\left(\sum_{1\le i\le n} D_i\right)^2}{\kappa^2}\log\frac{1}{\varepsilon \parallel v^{(0)} \parallel} \tag{8.58}$$

Proof Note that the number of iterations required for convergence is $O\left(\frac{1}{\log 1/\rho(M)}\right)$. We will prove the result by bounding $\rho(M)$.

We convert the problem into a problem involving a Markov chain. We augment M to give a row-stochastic matrix P:

$$P_{ij} = \begin{cases} M_{ij} & \text{if } 1 \le i, j \le n \\ 1/D_i & \text{if } j = n+1, i \le n, \text{ and } i \text{ is a neighbor of } z \\ 1 & \text{if } i = j = N+1 \\ 0 & \text{otherwise} \end{cases} \tag{8.59}$$

One can interpret P as providing the update equation for the reference node also:

$$\begin{pmatrix} \bar{v}^{(k+1)} \\ 0 \end{pmatrix} = P \begin{pmatrix} \bar{v}^{(k)} \\ 0 \end{pmatrix} \tag{8.60}$$

The corresponding Markov chain has a communicating class of n states and a single absorbing state corresponding to the reference node. Its unique steady-state distribution is therefore $[0, 0, \ldots, 1]$.

If λ is an eigenvalue of M, with eigenvector v, then it is also an eigenvalue of P since

$$P\begin{pmatrix} v \\ 0 \end{pmatrix} = \begin{pmatrix} Mv \\ 0 \end{pmatrix} = \lambda \begin{pmatrix} v \\ 0 \end{pmatrix} \tag{8.61}$$

Hence $\rho(M) \le |\lambda_2|$, where λ_2 is the second largest eigenvalue of P in magnitude. So we only need to bound λ_2.

Since our Markov chain is absorbing and not irreducible, to make use of the results of [22], we define a new Markov chain with transition probability matrix P^ε, where $0 < \varepsilon < 1/D_0$, by only altering the transition probabilities from the reference node: the probability of transitioning to every neighbor of the reference node is ε, with a self-loop back to the reference node having probability $1 - D_0\varepsilon$. Now P^ε irreducible and aperiodic. Its invariant distribution π satisfies

$$\varepsilon\pi_0 = \pi_i/D_i, \text{ for every neighbor } i \text{ of the reference node, and}$$
$$\pi_i/D_i = \pi_j/D_j, \text{ for neighboring nodes } i, j \text{ that are not the reference node.}$$

Hence,

$$\pi_0 = \frac{1}{1+\varepsilon\left(\sum_{1\le j\le n} D_j\right)}, \pi_i = \frac{\varepsilon D_i}{1+\varepsilon\left(\sum_{1\le j\le n} D_j\right)}$$

for $1 \le i \le n$. It also follows that P^ε is reversible, since $\pi_i P^\varepsilon_{ij} = \pi_j P^\varepsilon_{ji}$.

Now we employ Cheeger's inequality [22] to bound λ_2. Define the matrix Q by $Q(i,j) = P^\varepsilon_{ij}\pi_i = P^\varepsilon_{ij}\pi_j$. Denote $\pi(S) := \sum_{i\in S}\pi_i$, and $Q(S\times S^c) = \sum_{i\in S, j\in S^c} Q_{ij}$. Let

$$h := \min_{\pi(S)\le 1/2} \frac{Q(S\times S^c)}{\pi(S)} \tag{8.62}$$

By Cheeger's inequality,

$$1-2h \le \lambda_2 \le 1-h^2 \tag{8.63}$$

So now we only need to bound h.

Using $S = \{1, 2, \ldots, n\}$ in (8.62) gives the upper bound

$$h \le \frac{D_0}{\sum_{1\le i\le n} D_i} \tag{8.64}$$

and so

$$\lambda_2 \ge 1 - 2\frac{D_0}{\sum_{1\le i\le n} D_i} \tag{8.65}$$

Any set S as in (8.62) cannot contain the root node, since its probability would then exceed 1/2. So $S = \{1, 2, \ldots, n\}$ maximizes the denominator of the right-hand side of (8.62). Hence

$$\pi(S) = \sum_{1\le i\le n}\pi_i = \frac{\varepsilon\left(\sum_{1\le j\le n} D_j\right)}{1+\varepsilon\left(\sum_{1\le j\le n} D_j\right)} \tag{8.66}$$

Also, for every i, j,

$$Q(i,j) = \frac{\varepsilon}{1+\varepsilon\left(\sum_{1\le j\le n} D_j\right)} \tag{8.67}$$

Furthermore, the minimum size of the cut $S \times S^c$ is equal to the edge-connectivity κ of the graph by definition. Hence, numerator of RHS of (8.62) $\geq \frac{\kappa\varepsilon}{1+\varepsilon\left(\sum_{1\leq j\leq n} D_j\right)}$. Therefore

$$h \geq \frac{\kappa}{\sum_{1\leq i\leq n} D_i} \tag{8.68}$$

and so

$$\lambda_2 \leq 1 - \left(\frac{\kappa}{\sum_{1\leq i\leq n} D_i)}\right)^2 \tag{8.69}$$

Since eigenvalues are continuous functions of a matrix (see [23], Appendix 1), and $P^\varepsilon \to P$,

$$\lim_{\varepsilon\to 0} \lambda_2^\varepsilon = \lambda_2 \tag{8.70}$$

Since neither the upper nor the lower bounds derived above depend on ε,

$$1 - 2\frac{D_0}{\sum_{1\leq i\leq n} D_i} \leq \lambda_2 \leq 1 - \left(\frac{\kappa}{\sum_{1\leq i\leq n} D_i}\right)^2 \tag{8.71}$$

and the result follows by taking logarithms. ■

We can apply Theorem 6 to some of the canonical graphs considered earlier. For the lattice network, it gives a $O(1-1/n^2)$ bound on the second largest eigenvalue, which translates to a $O(n^2)$ upper bound on convergence time. The same $O(n^2)$ bound holds with high probability as $n \to \infty$ for the random planar network.

8.10 Decomposition Techniques to Speed Up Convergence

The global iterative algorithm described in the last section converges to the optimal least-squares solution. However, the convergence time is potentially a significant bottleneck to performance. This is for two reasons. First, a large convergence time in effect means a large number of messages to be passed. Each iteration of the synchronous algorithm requires each node to broadcast a message to its neighbors, meaning that an $\Omega(n)$ convergence time (which is a lower bound for a typical graph like a lattice) translates to $\Omega(n^2)$ messages in total, which is a significant overhead. The second and more important reason is that, in reality, the clock offsets vary with time (see [13]), and so the edge offset estimates are tracking a time-varying set of parameters. This in turn implies that for the node offset estimates to be reasonably accurate, their computation from the edge estimates (which is done via the iterative algorithm) must be more rapid than the variation time scales.

We now investigate some structural properties of the graph which could be exploited to speed up the convergence time of the algorithm. These properties allow the graph to be decomposed or split into components such that the optimal node estimates on each of the components can be combined in a simple manner to produce the optimal estimate over the entire graph. Our approach exploits the connection with electrical resistance: these methods of decomposition are also those which allow the resistance of the entire network to be computed by combining together resistances of the decomposed constituents. We call these methods of decomposition *series* and *parallel splitting*.

Series splitting is based on the following simple procedure. Consider a connected network graph G with root node 0. We wish to compute the optimal (minimum variance unbiased) estimate of the clock offset of a node i with respect to node 0, given edge estimates for all the edges in G. Now suppose that there is a *cut-vertex* between 0 and i, i.e., a node j such that the induced graph on $V(G) - j$ (where $V(G)$ is the vertex set of G) is disconnected into two components containing 0 and i, respectively. Let G_1 be the induced subgraph on the nodes of the first component along with j, and G_2 be the induced subgraph on the nodes of the second component along with j.

Now, let $v_{j0}^{(1)}$ be the least-squares estimate of the offset between j and 0, *given only the edges in* G_1, and let $v_{ij}^{(2)}$ be the least-squares estimate of the offset between i and j, *given only the edges in* G_2. Each of these least-squares estimates could be computed on the smaller subgraphs G_1 and G_2 by whatever technique is appropriate. Then, we set the estimate of the offset of i from 0 as $v_{j0}^{(1)} + v_{ij}^{(2)}$.

Lemma 4 *Let $\hat{v}_i$ be the least-squares optimal estimate of the offset between i and* 0 *for the graph G. Then, $\hat{v}_i = v_{j0}^{(1)} + v_{ij}^{(2)}$.*

Proof The proof of this lemma is quite simple. The estimate $v_{j0}^{(1)} + v_{ij}^{(2)}$ is unbiased because each of $v_{j0}^{(1)}$ and $v_{ij}^{(2)}$ is unbiased, and the true value v_i is the sum of the true offsets across $0j$ and ji. Furthermore, G_1 and G_2 are edge-disjoint, which means that the errors of each of the estimates $v_{j0}^{(1)}$ and $v_{ij}^{(2)}$, which are linear combinations of the corresponding edge estimates on the edges of G_1 and G_2, respectively, are also independent. Therefore, the variance of the error of $v_{j0}^{(1)} + v_{ij}^{(2)}$ is the sum of the variances of the two estimates. But the resistance between i and 0 in the graph G is the sum of the resistances R_{j0} and R_{ij} over G_1 and G_2, respectively, because the two components are in series. Therefore,

$$\begin{aligned}\operatorname{var}\left(v_{j0}^{(1)} + v_{ij}^{(2)}\right) &= \operatorname{var}\left(v_{j0}^{(1)}\right) + \operatorname{var}\left(v_{ij}^{(2)}\right)\\ &= \sigma^2 R_{j0} + \sigma^2 R_{ij}\\ &= \sigma^2 R_{i0}\end{aligned}$$

Hence, $v_{j0}^{(1)} + v_{ij}^{(2)}$ is also a minimum variance estimate. By the uniqueness of the minimum variance unbiased estimate shown in Theorem 1, $\hat{v}_i = v_{j0}^{(1)} + v_{ij}^{(2)}$. ■

We call the above operation *series splitting*.

Similarly, *parallel splitting* can be defined as follows. Consider a node i in a connected graph G with root node 0. If the induced graph on $V(G) - \{i, 0\}$ has more than one component, we say G can be *parallel split* between i and 0. For simplicity, suppose that the resulting graph has only two components (in general, it could have more than two). Define G_1 as induced subgraph on the nodes of the first component along with i and 0 and G_2 as the induced subgraph on the nodes of the second component along with i. Let R^1_{i0}, R^2_{i0}, and R_{i0} be the resistances between i and 0 on the graphs G_1, G_2, and G, respectively. By the parallel combination of resistances, we know that $1/R_{i0} = 1/R^1_{i0} + 1/R^2_{i0}$.

Let $v^{(1)}_{i0}$ be the least-squares estimate of the offset between i and 0 on G_1 and $v^{(2)}_{i0}$ be the least-squares estimate on G_2. Then, we set the estimate of the offset of i from 0. We then have the following simple result.

Lemma 5 *Let $\hat{v}_i$ be the least-squares optimal estimate of the offset between i and* 0 *for the graph G. Then, $\hat{v}_i = \frac{R^2_{i0}}{R^1_{i0}+R^2_{i0}} v^{(1)}_{j0} + \frac{R^1_{i0}}{R^1_{i0}+R^2_{i0}} v^{(2)}_{ij}$.*

Proof The proof is similar to that of Lemma 4. The estimate is unbiased since we are taking a convex combination of the two unbiased estimates $v^{(1)}_{i0}$ and $v^{(2)}_{i0}$. Since G_1 and G_2 are edge-disjoint, the variance of the linear combination is $\left(\frac{R^2_{i0}}{R^1_{i0}+R^2_{i0}}\right)^2 \operatorname{var}\left(v^{(1)}_{i0}\right) + \left(\frac{R^1_{i0}}{R^1_{i0}+R^2_{i0}}\right)^2 \operatorname{var}\left(v^{(2)}_{i0}\right)$, which equals $\sigma^2 \frac{R^1_{i0} R^2_{i0}}{R^1_{i0}+R^2_{i0}}$. But this is the resistance between i and 0 on G times σ^2, which means that this is the minimum variance estimate. The result follows. ∎

The above techniques provide ways to speed up the iterative synchronization procedure if the network graph has a particular structure which can be exploited. For example, if a graph can be efficiently split into two parts of approximately equal size, the iterative algorithms could be carried out in parallel for each subgraph and the results could then be combined according to the method of splitting, thus yielding about a factor of 2 improvement.

In general, the graph itself may not have such a structure in its entirety, but may instead have such a structure embedded within itself. It may then be possible to achieve some of these savings by only working on an embedded decomposable subgraph, which corresponds to throwing away some of the edge estimates. An extreme example is that of using a spanning tree, which we have discussed before. A slightly better approach would consist of combining the summed estimates over multiple edge-disjoint paths in inverse proportion to their length, which would reduce the error variance by parallelism. Finding topology control techniques to select such subgraphs for decomposition is an interesting problem in itself and worth investigating.

Another possibility is to find a subgraph which can be recursively split in series or in parallel. That is, the graph is initially split into components, each of which is further split into further components in series or in parallel, and so on. The complexity of such an approach would be that a substantial amount of global connectivity information would have to be maintained, because in each parallel split there would be a node which would have to combine the estimates of the components by a linear combination, depending on the corresponding resistances, which would have to be known.

8.11 Conclusion

This chapter conducts an investigation into the problem of clock synchronization in wireless sensor networks. We present an analysis of the spatial smoothing clock synchronization approach based on least-squares optimization and the corresponding distributed two-step synchronization algorithm. We have analyzed both asymptotic accuracy and the convergence time and its dependence on graph parameters and size.

We have restricted attention to the case where there is no clock drift, i.e., the skews of clocks are constant over time. If the skews do change with time, then one needs a stochastic model of clocks. This is treated in [13].

Acknowledgments This material is based upon work partially supported by NSF under contract nos. CNS-07-21992 and CNS-0626584, the USARO under contract no. W-911-NF-0710287, and AFOSR under contract no. FA9550-09-0121.

References

1. K. Plarre and P. Kumar. Object tracking by directional sensors. In *Proceedings of IEEE Conference on Decision and Control (CDC)*, Seville, Spain, pages 3123–3128, 2006.
2. A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 88–97, 2002.
3. S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *SenSys '03: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, New York, NY, pages 138–149, 2003.
4. M. Sichitiu and C. Veerarittiphan. Simple, accurate time synchronization for wireless sensor networks. In *IEEE Wireless Communications and Networking Conference(WCNC)*, New Orleans, LA, pages 1266–1273, 2003.
5. R. Karp, J. Elson, D. Estrin, and S. Shenker. Optimal and global time synchronization in sensornets, *Center for Embedded Networked Sensing*, Technical Reports, University of California, Los Angeles, April 2003.
6. R. Solis, V. Borkar, and P. R. Kumar. A new distributed time synchronization protocol for multihop wireless networks. *Technical Report*, University of Illinois, April 2005.
7. A. Giridhar and P. R. Kumar. Distributed clock synchronization over wireless networks: algorithms and analysis. *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, pages 4915–4920, December 13–15, 2006.
8. A. Giridhar. In-Network computation in wireless sensor networks. *Ph. D. Thesis*, Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign. April 2006.
9. B. Sundararaman, U. Buy, and A. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad-Hoc Networks*, 3(3): 281–323, May 2005.

10. S. Graham and P. R. Kumar. Time in general-purpose control systems: the control time protocol and an experimental evaluation. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, pages 4004–4009, December 14–17, 2004.
11. N. M. Freris and P. R. Kumar. Fundamental limits on synchronization of affine clocks in networks. In *Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, pages 921–926, December 12–14, 2007.
12. O. Gurewitz, I. Cidon, and M. Sidi. One-way delay estimation using network-wide measurements. *IEEE/ACM Transactions on Networking*, 14:2710–2724, June 2006.
13. N. M. Freris, V. S. Borkar, and P. R. Kumar. A model-based approach to clock synchronization. In *Proceedings of 48th IEEE Conference on Decision and Control*, Shanghai, December 16–18, 2009.
14. P. Gupta and P. R. Kumar. Critical power for asymptotic connectivity in wireless networks. In W. McEneaney, G. Yin, and Q. Zhang, *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming*, editors pages 547–566, Birkhauser, Boston, MA, 1998.
15. H. Poor. *An Introduction to Signal Detection and Estimation*, 2nd ed., Springer-Verlag, UK, 1995.
16. L. Chua, C. Desoer, and E. Kuh. *Linear and Nonlinear Circuits*. McGraw Hill Book Company, New York, NY, 1987.
17. A. Jadbabaie. On geographical routing without location information. In *Proceedings of IEEE Conference on Decision and Control (CDC)*, Nassau, Bahamas, pages 4764–4769, 2004.
18. F. Y. Wu. Theory of resistor networks: The two-point resistance, *Journal of Physics A*, 37:6653, 2004.
19. F. Xue and P. R. Kumar. The number of neighbors needed for connectivity of wireless networks. *Wireless Networks*, 10(2):169–181, 2004.
20. D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1989.
21. C. D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, PA, 2000.
22. P. Diaconis and D. Stroock. Geometric bounds for eigenvalues of Markov chains. *Annals of Applied Probability*, 1(1):36–61, 1991.
23. R. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1990.
24. M. Penrose. Random geometric graphs. Oxford University Press, Oxford, UK, 2003.
25. J. Tsitsiklis. Problems in decentralized decision making and computation. *Ph.D. Dissertation*, Massachusetts Institute of Technology, 1984.

Chapter 9
Algorithmic Aspects of Sensor Localization

Sajal K. Das, Jing Wang, R.K. Ghosh, and Rupert Reiger

Abstract Identifying locations of nodes in wireless sensor networks (WSNs) is critical to both network operations and most application level tasks. Sensor nodes equipped with geographical positioning system (GPS) devices are aware of their locations at a precision level of few meters. However, installing GPS devices on a large number of sensor nodes not only is expensive but also affects the form factor of these nodes. Moreover, GPS-based localization is not applicable in the indoor environments such as buildings. There exists an extensive body of research literature that aims at obtaining absolute locations as well as relative spatial locations of nodes in a WSN without requiring specialized hardware at large scale. The typical approach consists of employing only a limited number of anchor nodes that are aware of their own locations, and then trying to infer locations of non-anchor nodes using graph-theoretic, geometric, statistical, optimization, and machine learning techniques. Thus, the literature represents a very rich ensemble of algorithmic techniques applicable to low power, highly distributed nodes with resource-optimal computations. In this chapter we take a close look at the algorithmic aspects of various important localization techniques for WSNs.

9.1 Introduction

A common vision on applications of wireless sensors is a large number of distributed devices embedded for tight interactions with physical world. An essential challenge to work with a wireless sensor network (WSN) is to create a seamless coupling with the physical world, often as an unattended control system. In order to operate as a long-lived unattended control system, a WSN must be energy aware, self-configuring, and should achieve desired global behavior by predominantly localized algorithms. Understandably, a network-wide collaboration is absolutely essential in

S. K. Das (✉)
Department of Computer Science and Engineering, Center for Research in Wireless Mobility and Networking (CReWMaN), University of Texas at Arlington, Arlington, TX 76019, USA
e-mail: das@uta.edu

order to achieve expected global behavior. For meaningful collaborative processing, individual sensor data should be associated with spatial and temporal coordinates. Therefore, energy-efficient techniques for sensor localizations are critical to the operations of a WSN as an unattended system.

9.1.1 Importance of Localization

Sensors are used for gathering environmental data such as temperature, pressure, humidity, radiosity. The collected data assist in predicting likely occurrence of events such as bush fire, radio-active leaks, failures in structures, and many other impending disasters including earthquakes, floods, and weather changes. Early prediction of such events helps in planning adequate response system that may either prevent those events or mitigate the consequential damages. The response system should have the ability to extract context from the gathered sensory data if it were to predict the events correctly and operate at the expected level of efficiency. A context is defined by ambiences among which location (both spatial and temporal) is the most critical ingredient.

WSNs with sensing capabilities can gather vital security-related parameters such as radio communication, signs of accelerated activities, or vigorous movements in an area to aid in developing a security response and advanced warning system. But these sensed parameters are useless unless they are accompanied by corresponding location information. Although location information can be fed manually, it may not be feasible in a large deployment area . So it is necessary to develop a system that can automatically update location information of all nodes in the deployment area.

Navigation and vehicle tracking is another area where the use of WSNs is found to be extremely useful. Vehicle tracking with autonomous interception mechanism can be deployed in an outdoor area. It senses entry as well as movement of an offending evader in the area. A cooperative mobile agent may be dispatched for intercepting the evader as soon it gets detected before any damage is done. The successful realization of such a tracking and interception system is dependent on the location information in two tiers. First, the sensors must be able to detect the evader as soon as it enters the area and be able to track it while the evader continues to move around in the area under observation. Second, the update on the location information of the evader must be routed to intercepting mobile agent so long as it continues to pursue the evader.

Apart from context-related computations in application level tasks discussed above, the knowledge of sensor locations is also essential for network level operations in WSNs. In a WSN environment, some nodes may die out due to fast drainage of battery. Often, in order to reestablish coverage and connectivity, new nodes may also be injected into the network, or old dead nodes may become alive following battery replacements. Under this scenario of frequent changes in topology of WSNs, geography routings are found to be more efficient than topology-based routing schemes. The basic issue that should be addressed in a geography routing scheme is its ability to gather location information and to have a location tracking mechanism for establishing connectivity before routing data. Resource scarcity prohibits sensors to maintain tables of locations. Moreover, the dynamic nature of topology configuration renders location tables frequently useless. So, localization or finding locations of sensor nodes is a fundamental step in routing or transmission of data in a WSN.

The problem of localization in WSNs attracted a lot of attention from researchers. A compendium of knowledge representing rich ensemble of algorithmic issues can be found in the existing literature. In this chapter, we attempt to review the existing literature with emphasis on the algorithmic roots of sensor localization techniques.

9.1.2 Generic Approach to Solution

Most localization schemes approach the problem with the assumption that there is a small set of nodes which are aware of their locations. Such nodes, typically known as *anchors*, are placed at fixed points or equipped with geographical position system (GPS) devices. The anchor nodes serve as references for the localization of other nodes. The number, the density, and the deployment of reference nodes raise a trade-off with the location accuracies achievable through localization methods based on proximity to anchor nodes.

In order to alleviate the inherent problem of low accuracy in proximity-based localization, additional measurements on distance between nodes or angles or combined measurements have been proposed. Distance measurements can be obtained by utilizing the radio signal strength indicator (RSSI), time of arrival (TOA), time difference of arrivals (TDoAs), and so on, while angle measurements rely on compasses or radio array. Given the reference points and the spatial relationship, nodes can be localized by exploiting the geometric relationships among them. The trade-off between hardware cost and the accuracy of measurements motivates the study on probabilistic model for radio signal strength leading to localization results in terms of the distribution of locations or deployment regions.

Absolute location can be determined if the anchor nodes are aware of their absolute locations. However, in certain applications absolute locations may not be very important. Localization methods with the absence of known locations focus on the problem of forming a map of nodes with respect to a stand-alone coordinate system [1]. Given the absolute locations of a subset of the nodes, these relative locations can be transformed into absolute ones when needed.

The use of mobile reference nodes is advantageous for localization of non-localized nodes as it becomes possible to get more measurements on spatial relationships by moving anchors along certain trajectories. A mobile reference node likely to have more resources compared to the ordinary static nodes that need to be localized. However, the cost of deploying a mobile anchor can be high. So, only a limited number of mobile reference nodes can be deployed. Yet mobile-assisted localization is able to bring significant improvement to the localization of static nodes.

9.1.3 Known Algorithmic Approaches

The localization methods proposed for sensor nodes can be categorized into five groups: distributed algorithms, centralized algorithms, iterative algorithms, mobility-assisted approach, and statistical techniques.

In distributed algorithms, a sensor node localizes itself through message exchange with a number of neighbors who are aware of their locations. It is essentially the propagation of the location information of the anchor nodes, which know their own locations, through interactions among anchor nodes and non-anchor nodes. The anchor nodes can be more than one hop away from the non-anchor nodes when the density of the anchor nodes is low. Less is the number of anchors, the more the localization relies on the spatial relationship among non-anchor nodes. A distributed algorithm considers a part of the network when localizing one sensor node. The densities of anchor and non-anchor nodes play important roles in determining the size and topology of the partial network. Since the spatial relationship among sensor nodes can be inaccurate, a partial network consisting of one-hop neighbors may not be sufficient for estimating accurate locations. However, a partial network consisting of multihop neighbors may introduce more noises into the localization process. Therefore, the key point in the design of distributed algorithms is to obtain the most accurate location estimates given the densities of the nodes and the means of interpreting the spatial relationship.

When the density of the nodes and the information on their spatial relationship no longer suffice for good estimates of locations, a centralized algorithm may be more appropriate to use. It aims at obtaining locations of the non-anchor nodes with the network-wide message exchange and computations. A centralized algorithm is usually expensive to implement, although it naturally solves the anchor-free localization problem by providing a network-wide relative locations of the sensor nodes. It is worth noticing that a centralized algorithm can be transformed into a distributed algorithm by applying the centralized algorithm on the set of subnetworks that partitions the whole network.

The goals of iterative techniques include the control of error propagation and stitching local maps of sensor nodes. A centralized algorithm can be applied iteratively on clusters of sensor nodes. Whereas, a distributed algorithm adopts iterative techniques to refine the initial estimate on the locations through feeding more range measurements or information of spatial relationship to the localization process.

In order to deal with low density of anchor nodes, mobile anchors are introduced to improve the localization accuracy. Furthermore, there exist approaches that take advantage of the mobility of sensor nodes. These mobility-assisted approaches are mostly distributed algorithms due to the dynamic nature of the scenario.

Statistical techniques have been proposed to localize sensor nodes when other approaches tend to fail because of the noisy measurements, especially in indoor environments. They are a mix of centralized and distributed algorithms and also rely on the iterative process to refine the location estimates.

From the above discussions, we discern that the classification of distributed algorithms and centralized algorithms provides only little indication about the features of the localization approaches. Therefore, it is more natural to study localization algorithms under four groups, namely, range-free, range-based, anchor-free, and anchor-based. In addition, we also discuss the iterative process, the mobility-assisted approach, and the statistical techniques.

9.1.4 Inherent Challenges

The inherent challenges facing the localization approaches include the network density, the noisy measurements, and the resource constraints. The first two challenges inspire a variety of approaches of different algorithmic features, while the last challenge corresponds to the complexity of the algorithm applied in the approach.

Network Density: Localization algorithms are required to deal with different densities of sensor nodes. In a dense network with abundant reference nodes, a good localization result could be accomplished without much difficulty. In contrast, for sparse networks, the available knowledge, such as the known locations, proximity information, range measurements, or angle measurements, could be insufficient for determining the exact locations of nodes. Thus, the key challenge for the localization problem in a sparse network is to achieve the maximum localization accuracy given a limited number of anchor nodes or measurements.

Noisy Measurements: Since measurements on proximity, range, and angle are subject to noise due to the inherent uncertainty of a wireless signal, localization algorithms are expected to be able to deal with noisy measurements. Therefore, additional efforts on modeling the noises and alleviating the impacts on localization performance are critical to the success of localization methods.

Resource Constraints: To enable cooperation among nodes in the localization process, information exchange between neighboring nodes adds to the energy consumption and the bandwidth occupancy. For centralized localization algorithms, where cooperation is orchestrated through a central node (usually the base station), extra communication cost is incurred for collecting and forwarding the measurements to the base stations and sending the localization results to the nodes.

A number of excellent surveys on localization in sensor networks have been published in recent years [2–5]. Most of them reported either the early results on localization in WSNs or methods having origin in cellular networks and robotics. Our focus in this chapter is not just yet another survey but to provide a comprehensive discussion on sensor node localization problem with representative references and an inherent perspective on the algorithmic aspects of localization approaches.

9.1.5 Chapter Organization

This chapter has been organized as follows. In Sect. 9.2, we study the algorithmic aspects of range-free techniques based on the unit disk model and other related graph theoretic solutions. Section 9.3 deals with range-based techniques which depend on geometry of nearness exploiting relationships among spatial coordinates of sensor nodes and anchor nodes. Some of the localization approaches rely on the use of additional hardware. These techniques are preferable, especially, if accuracy is the overriding concern. Section 9.4 provides a summary of such techniques. Optimization techniques work with iterative refinement of crude initial solutions. These techniques can, therefore, be clubbed as iterative processes. Section 9.5 deals with two important iterative techniques for localization. Section 9.6 addresses the issue of low density of anchor nodes by employing few resource-rich mobile reference nodes. Statistical techniques can be viewed as some sort of fingerprinting mechanism to infer localization by exploiting the relationship between distances and signal measurements. These techniques are discussed in Sect. 9.7. Section 9.8 deals with performance issues while Sect. 9.9 talks about open issues in sensor localization. Finally, Sect. 9.10 concludes this chapter.

9.2 Range-Free Localization

Sometimes, range measurements may not be available due to cost constraints. Under this situation, proximity information provided by the radios attached to the sensor nodes could lead to acceptable solutions for the localization problem. A wireless sensor node is usually represented by a unit disk model to formulate localization as graph theoretic problem. The localization is accomplished through a graph composed of vertices representing the sensor nodes and edges representing the radio link between the nodes. Anchor nodes (also known as reference nodes, or beacon nodes, or landmarks), deployed at fixed locations or equipped with GPS devices, can feed known locations to the localization process. The location information then propagates to the other nodes according to spatial relationships among anchor and non-anchor nodes and also between non-anchor nodes. More precisely, the network is represented by a graph $G(V, E)$. A subset of nodes $H = \{v_1, v_2, \ldots, v_m\} \subseteq V$ are aware of their respective locations $(r_1, r_2, \ldots, r_m) \in \Re^d$. The proximity measurements are represented using the adjacency matrix and the distance matrix. The goal is to obtain the estimation of the locations $(s_1, s_2, \ldots, s_{n-m}) \in \Re^d$ of the remaining set of nodes $V - H$.

Figure 9.1 illustrates an example of the above model. Five sensor nodes, whose locations are denoted as s_1, s_2, s_3, s_4, and s_5, need to be localized with the help of

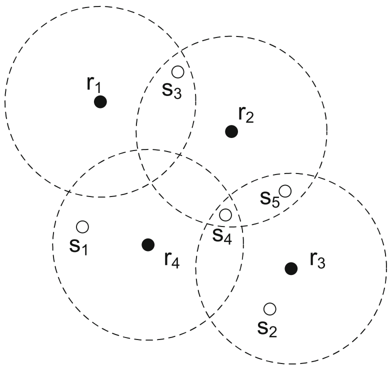


Fig. 9.1 An example of localization

four reference nodes with known locations r_1, r_2, r_3, and r_4. The adjacency matrix and the distance matrix can be obtained given the proximity information collected at the sensor nodes and the anchors. Note that different sensors would have different number of reference nodes in their proximity. The accuracy of location estimates for the unlocalized sensors increases as a function of the number of reference nodes in the neighborhood.

9.2.1 Anchor-Based Approaches

Given densely deployed anchors in the network, the locations of sensor nodes can be estimated with the help of the known locations of the anchors. A simple solution to the proximity-based localization is to determine the *Centroid*, as proposed in [6]. It computes a node's location approximated by the centroid of the locations of anchors in its proximity. Consider the example shown in Fig. 9.1. The estimated locations of nodes s_1, s_2, s_3, s_4, and s_5 can be obtained as follows using the centroid technique:

$$\begin{aligned}
(x_{s_1}, y_{s_1}) &= (x_{r_4}, y_{r_4}) \\
(x_{s_2}, y_{s_2}) &= (x_{r_3}, y_{r_3}) \\
(x_{s_3}, y_{s_3}) &= \left(\frac{x_{r_1} + x_{r_2}}{2}, \frac{y_{r_1} + y_{r_2}}{2}\right) \\
(x_{s_4}, y_{s_4}) &= \left(\frac{x_{r_2} + x_{r_3} + x_{r_4}}{3}, \frac{y_{r_2} + y_{r_3} + y_{r_4}}{3}\right) \\
(x_{s_5}, y_{s_5}) &= \left(\frac{x_{r_2} + x_{r_3}}{2}, \frac{y_{r_2} + y_{r_3}}{2}\right)
\end{aligned}$$

The accuracy of the centroid method relies heavily on the density of anchors. Low anchor density results in the deterioration in performance. The problem of low anchor density can be tackled through simple modifications to the centroid method. The underlying idea is to incrementally increase the density of reference nodes by including freshly localized nodes into the localization process. The modified method also allows anchors residing several hops away to be involved in localizing the nodes. However, this approach leads to the propagation of localization errors along with the location information. Weighted centroid or confidence-based centroid were introduced to address the problem of restricting error propagation in modified centroid methods [7, 8]. Localized nodes are assigned different weights or confidence levels in order to counterbalance the accumulation of location error from the localized nodes acting as anchors.

To avoid the accumulation of location error in propagating the location information of anchor nodes, geometric characteristic of the spatial relationship among sensor nodes has been adopted in the graph theoretic techniques. An example is presented in *APIT* [9], in which the location of a non-anchor node can be inferred from the region it could possibly reside in. As shown in Fig. 9.2, each non-anchor node runs the point in triangle (PIT) tests to find the triangle regions it resides in.

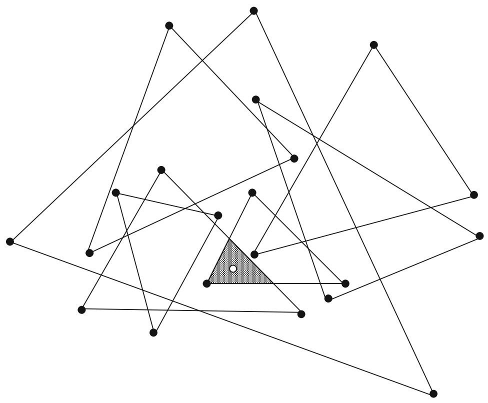


Fig. 9.2 Overview of APIT

Each triangle region is formed by obtaining locations of three non-collinear anchors. The location of the non-anchor node is estimated to be the center of gravity of the intersection of the triangles, where the non-anchor resides. For proximity-based localization of static nodes, it is hard for the non-anchor nodes to perform the PIT test. The authors in [9] presented an approximate PIT test, in which the node is only required to be able to determine if any one of its neighbors is farther/closer to all the three anchors forming its residing triangle. Therefore, the location error of APIT roots in the approximate PIT test. Its performance relies heavily on the density of the network, where it is suggested that the degree of connectivity should exceed 6.

The performance of proximity-based localization schemes depends on the positions of the anchor nodes. Intentional deployment of anchor nodes was exploited in [10] to divide the plane into location regions defined by the overlapping regions of sensing ranges of the anchors as shown in Fig. 9.3. Instead of broadcasting its

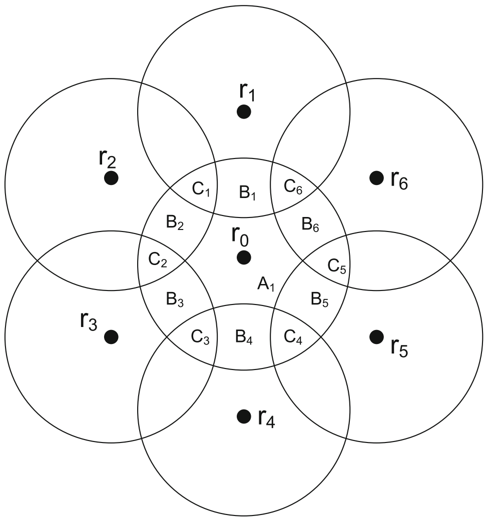


Fig. 9.3 Deployment of anchors

own position, an anchor node broadcasts the location regions consisting of its sensing range. For instance, the anchor node r_0 sends out beacons containing the type $\{A, B, C\}$ and the centroid of the location regions $\{A_1, B_1, \dots, B_6, C_1, \dots, C_6\}$, which are indeed the overlapping regions of the anchor nodes $\{r_0, r_1, \dots, r_6\}$. Upon receiving beacons from multiple anchors, the non-anchor node extracts the overlapping region of the anchors and takes the centroid of the overlapping region as the estimate on its own location. Compared to APIT and the modified centroid, the cell overlapping approach neither accumulates error in propagating the locations of anchors nor requires the RSSI value to perform the PIT test.

To overcome the problem of low density of anchors, *gradient* [11] and *DV-hop* [12] focus on localizing non-anchor nodes with the knowledge of radio range and locations of the anchors multiple hops away. The idea of Gradient is to estimate the distance between a pair of anchor and non-anchor nodes by multiplying hop count of the non-anchor nodes with the radio range. After obtaining distances to at least three anchors, the node applies multilateration algorithm to find out its own location. In contrast, DV-hop computes the average hop distance as anchors exchange their locations and hop count between them. After obtaining pairwise distances from a non-anchor node to an anchor by multiplying the average hop distance and the hop count from that anchor, triangulation is performed to estimate locations of the non-anchor nodes.

9.2.2 Anchor-Free Approaches

Multidimensional scaling (**MDS-MAP**) [13] tackles the localization problem without using anchors. It estimates locations with the proximity information and the radio range. The MDS approach includes three steps. The first step is to form the distance matrix with distances between all pairs of nodes in the network. In the absence of anchors, the distance is inferred from the multiplication of the hop count and radio range. Then, in the second step, the singular vector decomposition (SVD) is performed to determine an initial relative map of the nodes on the plane. The last step performs the necessary flip, rotation, and scaling according to the distances between anchors if there is any. Otherwise, the relative map would be the result of SVD. It was shown that the time complexities of the first two steps are both $O(n^3)$, where n is the number of nodes in the network. The MDS approach is also applicable when only a few anchor nodes are available.

9.3 Range-Based Localization

Geometric techniques manage to estimate the locations of the sensor nodes from the range measurements and geometric computations. The underlying idea is that Euclidean distance between two sensor nodes can be measured by their radio signals through RSSI, TOA, TDOA, etc. The presence of anchor nodes also plays an important role in geometric techniques in terms of the complexity of the problem and the difference between the localization goals: absolute versus relative locations.

9.3.1 Range Measurements

Since the nodes are equipped with radios to perform communications, the distance estimation by measuring the radio signal strength has attracted lot of attention [14]. A simplified model for RSSI-based range measurement is given by the following equation:

$$\mathrm{RSSI} \propto d^{-\alpha} \tag{9.1}$$

where d is the distance and α is a constant relevant to the environment. Given a RSSI value measured by the radio, the radio receiver is able to infer its distance from the sender. However, RSSI-based range measurement is extremely susceptible to noises and known for unreliability. An improvement on RSSI range measurements is proposed in [15] to reduce the distance estimation error by calibrating the range errors with RSSI values between known locations. Efforts have been made to obtain the mapping between RSSI measurements and the associated distances capturing the impacts of multipath fading, variations in temperature and humidity, human mobility, and changes in space layout on RSSI measurements in indoor localization [16]. The approaches based on the probabilistic model of RSSI range measurements are also introduced to address the uncertainties and irregularities of the radio patterns. For instance, a log-normal model was adopted in [17], which assumes that a particular RSSI value can be mapped to a log-normal distribution of the distance between the two nodes, as in (9.2).

$$\mathrm{RSSI} \longrightarrow \lg d \sim N(\mu, \sigma) \tag{9.2}$$

where d is the distance between the nodes, and $N(\mu, \sigma)$ is a normal distribution with mean μ and standard deviation σ.

Another common method for range measurement is based on the time difference of arrivals (TDOAs). It estimates the distance between the nodes from measurements on the time differences of arrivals of signals. The signal could be radio frequency (RF), acoustic, or ultrasound [18–20]. An example of utilizing TDOA was introduced in [21]. As shown in Fig. 9.4, the radio signal and ultrasound pulses are sent simultaneously. Given the time difference of the arrivals, the distance between the sender and the receiver can be obtained by multiplying the time difference and the speed of the ultrasound signal. Similarly, ranging techniques based on the time of arrival (TOA), which rely on capturing the signal's time of flight, obtain the distance by multiplying the time of flight with the signal speed [22]. The major challenge facing TOA-based ranging techniques is the difficulty of accurately measuring the

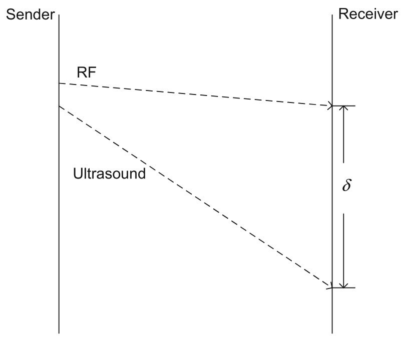


Fig. 9.4 An example of TDOA

time of flight, since the propagation speed could be extremely high compared to the distance to be measured.

9.3.2 Localization Problems Using Range Measurements

Given accurate range measurements, the localization algorithms are expected to produce the exact locations instead of raw estimations obtained from the proximity-based localization. The Euclidean distances between nodes can be interpreted both geometrically and mathematically. And the distance could be from one reference node to a non-localized node, denoted by $d_{i,j}$, or between two non-localized nodes, denoted by $\overline{d}_{i,j}$.

The locations of the nodes satisfy the following equations:

$$\| s_i - s_j \|^2 = \bar{d}_{i,j}$$
$$\| r_i - s_j \|^2 = d_{i,j}$$

Although computing the distances between each pair of locations is a trivial problem, the inverse problem, which tries to find locations of the nodes given the Euclidean distances between each pair of nodes is far from trivial. It can be formulated as a graph realization problem, aiming at mapping the nodes in the graph to points in the plane so that the Euclidean distances between nodes equal the respective edge weights. The fundamental problem in graph realization is the rigidity of the graph. For example, graphs in Fig. 9.5(a), (b) exhibit non-rigidity. Given a set of nodes and the Euclidean distances between each pair of nodes, the locations of nodes may not be unique. An example of a rigid graph is provided in Fig. 9.5(c), in which the nodes are uniquely localized given the distances. More discussions on graph rigidity and network localization can be found in [23].

The study in [24] proved that the localization with distance information in sparse networks is an NP-hard problem, while localization with distances of $\Omega(n^2)$ pairs of nodes can be solved in polynomial time [25].

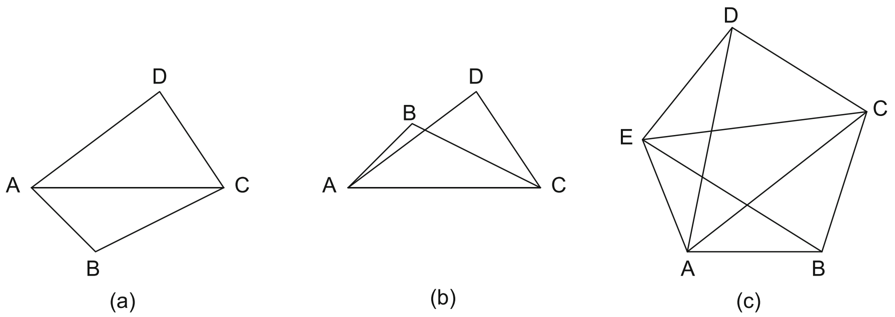


Fig. 9.5 Graph rigidity

9.3.3 Anchor-Based Approaches

Multilateration can be applied to obtain exact coordinates of a non-anchor node, given at least three anchors in the non-anchor node's proximity and the pairwise measurements between the anchor and non-anchor nodes. Assume that the coordinates of m anchors are available, and that the distances between nodes can be obtained through ranging techniques. Let d_{ij} denote the measurement of distance between the ith anchor and the jth non-anchor node. The multilateration problem concerning localization is then formulated as follows:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{9.3}$$

The estimation error E_j of the jth node is given by:

$$E_j = \sum_{i=1}^{m} (d_{ij} - \hat{d}_{ij})^2 \tag{9.4}$$

where $\hat{d}_{ij}$ is the estimated distance obtained by substituting the coordinates of the jth node with the estimated coordinates in (9.3). Gradient descent can be applied to obtain the coordinates of the jth node achieving the least-squared error. A similar approach was presented in [26].

The low density of anchors poses a challenge to the multilateration approach. In order to apply multilateration, *DV-distance* [27] follows the similar approach as *DV-hop* [12]. The distances of each hop are summed up to approximate the distance between a non-anchor node and an anchor node that is multiple hops away. The approximated distance is then used in the localization process. Instead of multihop distance approximation, the *Euclidean* method was proposed to compute the true distance through geometric relationships and the single hop distances. A detailed discussion on a localization protocol that follows the similar idea of Euclidean can be found in [28].

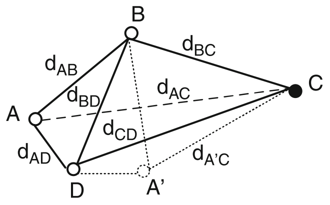


Fig. 9.6 Multihop localization

An example of multihop localization is demonstrated in Fig. 9.6. According to DV-distance, the distance between the non-anchor node A and the anchor node D can be approximated as $d_{AB} + d_{BC}$ or $d_{AD} + d_{CD}$ depending on a certain voting mechanism. In contrast, the Euclidean method manages to obtain the multihop distance d_{AC} by exploiting the geometric property of the quadrilateral $ABCD$. Although Euclidean focuses on computing the true distance to the anchor, it faces the rigidity problem due to low density of nodes. Given the set of single hop distances, the position of node A is not unique. As shown in Fig. 9.6, node A' leads to the same range measurements as node A. Additional neighbors and the corresponding range measurements are needed to eliminate the false estimation. According to [29], "an average of 11–12 degree of nodes in the ranging neighborhood" is required to have 90% of the network to be localized with a localization error of 5%. An iterative algorithm adopting the similar idea can be found in [30].

A trade-off between energy efficiency and location accuracy is unveiled in the multihop localization approaches facing low density of anchors. The cost of Euclidean is higher than DV-hop due to the collaboration among neighboring nodes in order to compute the true Euclidean distances. Whereas, DV-hop suffers from higher location error introduced by the approximation of multihop distances.

Although the semi-definite programming (**SDP**) approach [31] can be modified to incorporate the range measurements in the localization process by replacing the approximated distances with the measured distances, it tends to produce large localization error when the anchors are placed in the perimeter of the area. A different SDP problem is formulated in [32]. It manages to improve the localization performance by relaxing the equality constraint to the inequality constraint. The original problem of localizing n nodes using m anchors is formulated as follows:

$$\begin{aligned}
&\text{Find} && X \in \Re^{2\times n}, Y \in \Re^{n\times n} && \\
&\text{such that} && (e_i - e_j)^T Y (e_i - e_j) = d_{ji}^2 && \forall i, j \leqslant n \\
&\begin{pmatrix} \alpha_k \\ -e_j \end{pmatrix}^T && \begin{pmatrix} I_2 & X \\ X^T & Y \end{pmatrix} \begin{pmatrix} \alpha_k \\ -e_j \end{pmatrix} = d_{jk}^2 && \forall j \leqslant n, k \leqslant m \\
& && Y = X^T X &&
\end{aligned} \tag{9.5}$$

where α_k is the locations of the m anchors, e_i is a vector with zeros except the ith entry, d_{ji} is the distance between the ith and jth non-anchor nodes pair, and d_{jk} is the distance between the kth anchor node and the jth non-anchor node. The above localization problem can be transferred into a standard SDP feasibility problem by changing $Y = X^T X$ to $Y \geqslant X^T X$, which is equivalent to

$$Z = \begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} \succcurlyeq 0$$

The corresponding SDP problem is defined as follows:

$$\begin{aligned}
\text{Find} \quad & Z \in \Re^{(n+2)\times(n+2)} \\
\text{such that} \quad & (1;0;\mathbf{0})^T Z(1;0;\mathbf{0}) = 1 \\
& (0;1;\mathbf{0})^T Z(0;1;\mathbf{0}) = 1 \\
& (1;1;\mathbf{0})^T Z(1;1;\mathbf{0}) = 2 \\
& (0;e_i-e_j)^T Z(0;e_i-e_j) = d_{ji}^2 \qquad \forall i,j \leqslant n \\
& (\alpha_k;-e_j)^T Z(\alpha_k;-e_j) = d_{jk}^2 \qquad \forall j \leqslant n, k \leqslant m \\
& Z \succcurlyeq 0
\end{aligned} \tag{9.6}$$

In order to guarantee that the solution of SDP in (9.6) is indeed the solution to the original problem in (9.5), accurate distance measurements of $2n + n(n+1)/2$ pairs of nodes are required. Given noisy range measurements, the SDP problem can be formulated using the inequality constraints instead of the equality constraints on distances between nodes. The computation complexity of SDP is $O(n^3)$. Since it is expensive to apply the SDP method in localizing the whole network in a centralized process, a distributed SDP method was presented to address the scalability.

While plain MDS has been proposed in localization with connectivity information, modified MDS methods were proposed to localize neighboring nodes with range measurements. An iterative MDS approach was presented in [33] to deal with the absence of some pairwise distances. It differs from the classic MDS approach by introducing weights w_{ij} in the objective function as in (9.7):

$$\sigma(X) = \sum_{i<j} w_{ij}(\delta_{ij} - d_{ij}(X))^2 \tag{9.7}$$

where $\sigma(X)$ is the localization error with respect to the location vector X, δ_{ij} is the range measurements, d_{ij} is the distance computed from the location vector X, and w_{ij} is the weight for each pair. For the absent pairwise range measurements, the corresponding weights are set to 0, and the rest of the weights are set to 1. The location vector X is initialized using random values $X^{(0)}$. The localization result $X^{(k)}$ obtained from the kth round is fed to the $(k+1)$th round as the initial estimation on the location vector for the MDS process. The iteration stops when certain accuracy level is reached.

9.3.4 Anchor-Free Approaches

Considering the fact that using multilateration, non-anchor nodes could get localized relative to one another with a certain number of range measurements, *DV-coordinates* [27] promoted the idea of a two-stage localization scheme. During the first stage, neighboring nodes establish a local coordinate system according to the range measurements. Through registration with neighbors, the nodes transform their local coordinates into a global coordinate system in the second stage. Due to the insufficient overlapping or false overlapping between neighboring nodes, the performance of DV-coordinates suffers from error propagation in the second stage.

The idea of DV-coordinates was explored further in [34]. It led to *robust quad* which become the building blocks of the local coordinate system in order to avoid flip ambiguity. First, the clusters consisting of overlapping robust quads are formed to establish the local coordinate system. As shown in Fig. 9.5(c), the rigidity of the robust quad guarantees that two robust quads, ABCD and ABCE, sharing three vertices form a rigid subgraph with five vertices. The rigidity of the clusters is maintained by induction. Then, to mitigate the impact of noisy range measurements, a threshold on the minimum angle of the robust quad was introduced. With all these efforts, robust quad is able to significantly reduce the location error in comparison with other similar approaches.

Triangulation, as proposed in [1], is able to set up a local coordinate system with three nodes and their pairwise distances. A triangulation example is depicted in Fig. 9.7. Node A tries to localize its neighbors B and C. A defines its own position as the origin of the local coordinate system and C to be along its horizontal axis. The locations of A's neighbors are as follows:

$$\begin{aligned} x_C &= 0 \\ y_C &= d_{AC} \\ x_B &= d_{AB} \cos \angle BAC \\ y_B &= d_{AB} \sin \angle BAC \\ \angle BAC &= \arccos \frac{d_{AB}^2 + d_{AC}^2 - d_{BC}^2}{2 d_{AB} d_{AC}} \end{aligned}$$

Applying similar derivations, any node can transform its locations to the coordinate system of its neighbor with the knowledge of the locations in the two coordinate systems and the pairwise distances. Therefore, the rest of the network can adjust the locations to one particular local coordinate system. The propagation of one particular coordinate system involves high cost of collaboration among nodes, which is not favorable to the energy-deficient wireless sensor nodes.

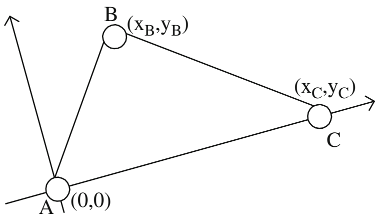


Fig. 9.7 Triangulation example

9.4 Techniques with Additional Hardware

Angle-based localization reduces the difficulty of the localization problem with knowledge on proximity or distances. However, the cost of applying angle measurement is remarkably high because of the antenna array or multiple receivers mounted on the nodes.

9.4.1 Angle Measurement

With the help of antenna array, it is possible for the nodes to measure the signal's angle of arrival (AOA) [35]. More recently, the angles between different edges of the connectivity graph can be obtained through multiple ultrasound receivers [36]. The AOA technique is adopted to assist both proximity-based and range-based localization as it is capable of improving the localization performance. The idea of angle measurement is demonstrated in Fig. 9.8, in which every node is able to measure the angle to its neighbor node and its own axis. For node A that is to be localized, it is aware of the two angles $\angle ab$ and $\angle ac$. The angle information can provide additional support to the localization or even localize the nodes solely based on the angle information [26].

9.4.2 Localization with Angle Measurement

The difficulty of localization solely on angle information has been studied in [37]. Fortunately, the angle information can be combined with the proximity-based localization. With angle information on all pairs of edges in the network, it is possible to transform the proximity-based localization problem from NP-hard to P class (polynomial-time solvable) irrespective of the number of anchor nodes available in the network. A more realistic scenario with angle information and knowledge on distances in a sparse network is further shown to be a problem in P, while it has been proved to be NP-hard to localize nodes in a sparse network solely with knowledge of the distances. Therefore, the AOA technique remains an attractive option for localization applications in spite of its cost and difficulty of deployment.

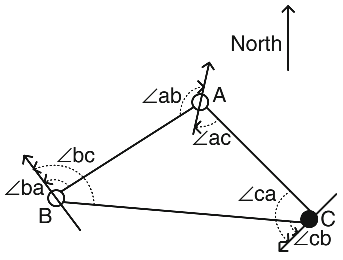


Fig. 9.8 Angle measurement

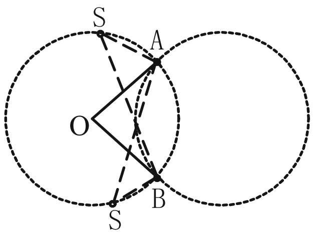


Fig. 9.9 Localization with two anchors

When nodes are enabled with AOA capability, localization using AOA can be reduced to multilateration by simple transformations. As presented in [26], given a number of anchors and the angle observations, non-anchor nodes obtain their locations through multilateration process. Examples of localization method using AOA are shown in Figs. 9.9 and 9.10. The anchors A and B are one-hop neighbors of non-anchor node S in Fig. 9.9. The location of S is confined to the dashed sector and the node S observes the AOA to A and B in terms of the angles $\angle ASB$ or $\angle BSA$. Alternatively, as shown in Fig. 9.10, a triplet of anchors can localize one-hop non-anchor node using the AOA measurement. The location of S is along the circumscribed circle for the anchors A, B, and C. The idea is to build a multilateration equation with the location of the non-anchor node, the center and the radius of the circle, which can be derived from the angle observed by the non-anchor node and the locations of the two or three anchors in the proximity of the non-anchor node. After obtaining a number of multilateration equations,

$$(x_S - x_{O_i})^2 + (y_S - y_{O_i})^2 = r_i^2 \tag{9.8}$$

the location of the non-anchor node can be computed with similar techniques for localization based on range measurements.

Although additional hardware is required for localization based on angle information, it is shown in [38] that the angle information can significantly reduce the difficulty of the localization problem based on solely range measurements. Besides, as shown in [29], the angle information can assist in localization by reducing the number of anchors and the network density in order to achieve satisfactory localization performance.

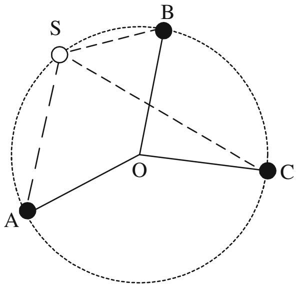


Fig. 9.10 Localization with three anchors

9.5 Techniques Based on Iterative Process

Local maps can be formed by running the iterative MDS over local range measurements. The alignment of the local maps is performed along the route from one anchor to another. In order to get absolute locations, at least three anchors are required to be present in the neighboring local maps. For further discussions on stitching the local maps, readers may refer to [39].

Forming clusters of nodes within the network has been regarded as an effective technique to improve performance of WSNs, especially for solving the scalability problem. Cluster-based local coordinate system has been proposed in [40] to set up a local coordinate system for a small subset of the network. The master nodes (cluster heads) are responsible for transforming the local coordinate systems into one global system. A counterpart of the cluster-based localization using only proximity information was presented in [41].

Another iterative technique is described in **dwMDS** [42]. It not only formulates the MDS problem with a novel optimization objective (the weighted cost function over multiple range measurements of pairwise distances) but also adopts an iterative algorithm starting from an initial estimation on the locations. Thus, the computational complexity is reduced from $O(n^3)$ in simple MDS to $O(nL)$, where n is the total number of nodes in the network and L is the number of iterations to satisfy a predefined accuracy level. dwMDS can also localize networks with no anchors by providing relative locations.

Since the error can be propagated and accumulated with the iterations, the error management scheme deserves further attentions in order to improve the performance of iterative methods. The potential error control techniques include selecting localized neighbors with certain level of accuracy in the iterative updates [43].

9.6 Mobility-Assisted Localization

Mobility of sensor nodes can be exploited in the localization process. For instance, an efficient sensor network design proposed in [44] took advantage of coverage overlaps over space and time because of the mobility of sensor nodes.

Mobility-assisted localization relieves WSNs from the significant cost of deploying GPS receivers and the pressure of provisioning energy for interacting with each other during the localization process. Anchor nodes equipped with GPS are capable of localizing themselves while moving. With the use of mobile anchor nodes, localization algorithms show significant savings on the installation cost and energy consumption and also improve accuracy.

In [45], an example scheme is proposed to localize static sensor nodes with one mobile beacon. As shown in Fig. 9.11, the mobile beacon periodically broadcasts beacon packets containing its coordinates while traversing the area where static sensor nodes are deployed. Upon receiving the beacon packets, a sensor node is able to infer its relative location to the beacon according to the radio signal strength (RSS)

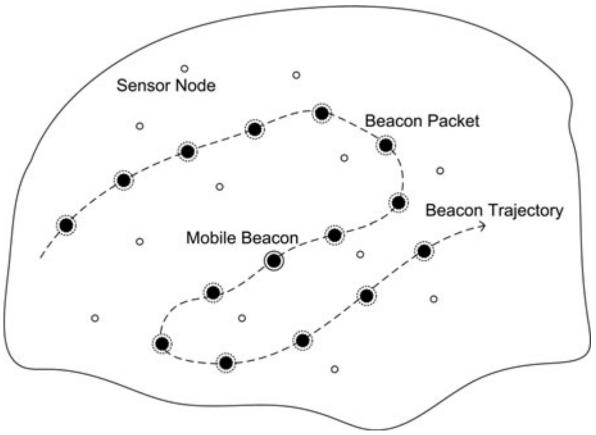


Fig. 9.11 Localization using a mobile beacon

of the beacon packet through Bayesian inference. The two key elements that are discussed by the authors [45] in their approach are calibration and beacon trajectory.

1. Calibration: In order to estimate the node's relative location to the moving beacon using the received RSS, it is necessary to calibrate the system, thus obtaining the propagation characteristic of the beacon packet in the air. The probability distribution function of the distance with respect to the received RSS is established given the calibration data.
2. Beacon trajectory: It determines the coverage of localization, the number of beacon packet broadcasts, and the localization accuracy. It is argued that the closer a beacon moves to the sensor node, the better is the localization accuracy. As a beacon trajectory can be regarded as a connected line of placements of static anchor nodes, it is quite obvious that non-collinearity and fold-freeness are also important to the beacon trajectory.

The computational complexity of the above localization algorithm is $O(n^2)$ and the storage requirement is also $O(n^2)$. As a result, it is possible to implement the localization algorithms on sensor nodes. Alternatively, the sensor nodes can report the RSS to a base station to avoid expensive computation with increased communication cost.

A similar method has been described in [46] as a special case. The idea is that each time a sensor node receives a beacon, it generates a quadratic constraint on its own location according to the radio range. Upon receiving all the beacon packets, the sensor node's location can be restricted in an intersected area of several bounding boxes as shown in Fig. 9.12.

Unlike the above GPS-based localization using mobile beacon, it was proposed in [47] to localize the sensor node using radio range of sensor node instead of that of mobile beacon. The method is based on a geometry conjecture, named perpendicular bisector of a chord, which states that a perpendicular bisector of a chord passes through the center of the circle (see Fig. 9.13 for an illustration).

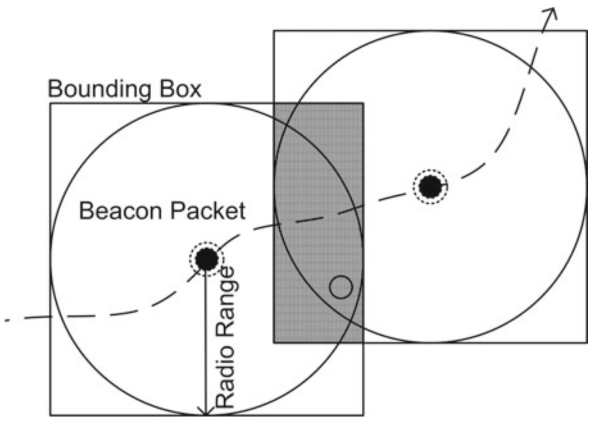


Fig. 9.12 Mobile beacon and bounding box

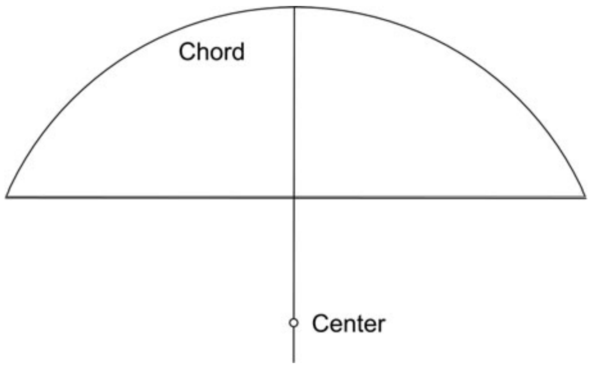


Fig. 9.13 Perpendicular bisector of a chord

Given the conjecture on the perpendicular bisector of a chord, it is possible to calculate a sensor node's location from its view of mobile beacon's movement. By maintaining a visitor's list, a sensor node is able to determine which beacon packets are sent when the mobile beacon is entering and leaving the sensor node's radio range. As shown in Fig. 9.14, the sensor node selects beacon packets to construct chords of its radio range and locate itself as the center of the circle. Their approach showed that range-free localization could also produce fine-grained location information with the help of mobile beacon's location broadcasts. The key factors affecting the localization performance are the following:

1. Beacon scheduling: Beacon packets are scheduled by adding a random jitter time to the period of beacon packet in order to avoid collisions among broadcasts from different beacons since the approach is not restricted to one mobile beacon.
2. Chord selection: As the probability of localization failures increases for short chords, a threshold of the length of the chord is applied to the selection of chords.
3. Radio range: Given a larger radio range, the localization error can be slightly reduced due to the larger threshold on the length of the chords.

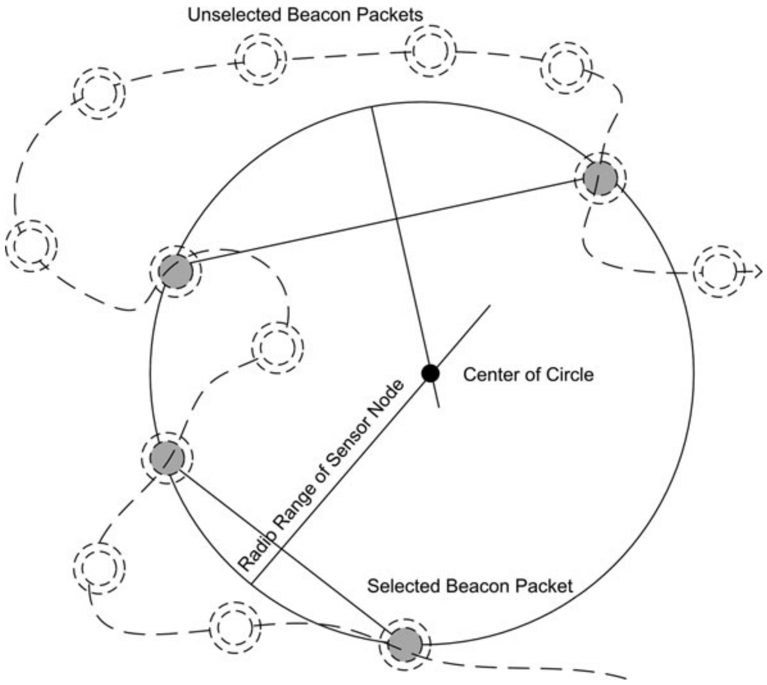


Fig. 9.14 Localization using intersect of perpendicular bisector of two chords

4. Beacon speed: Increased speed of mobile beacon movements can reduce the execution time of the localization because more beacon packets are sent during the same time period. However, it may also increase the localization error due to higher probability of selecting shorter chords.

9.7 Statistical Techniques

For indoor localization scenarios, the RSSI measurements suffer from severe multipath effects. Statistical approaches are regarded as promising candidates in dealing with noises and uncertainties of the measurements. The attempts to localize objects in the indoor environment with statistical techniques can be divided into two classes. One relies on mappings between the RSSI measurements and the locations, while the other manages to capture the statistical relationship between RSSI measurements and the distances. Both can work with off-line recording and on-line measurements in localizing the objects with RSSI measurement capability.

An example of mapping the RSSI profile of the space is *RADAR* [48], which involves two stages. As shown in Fig. 9.15, the RSSI values from multiple base stations (acting as anchors) are recorded at various locations during the first stage. Following this, a three-step localization process is performed in the second stage. In

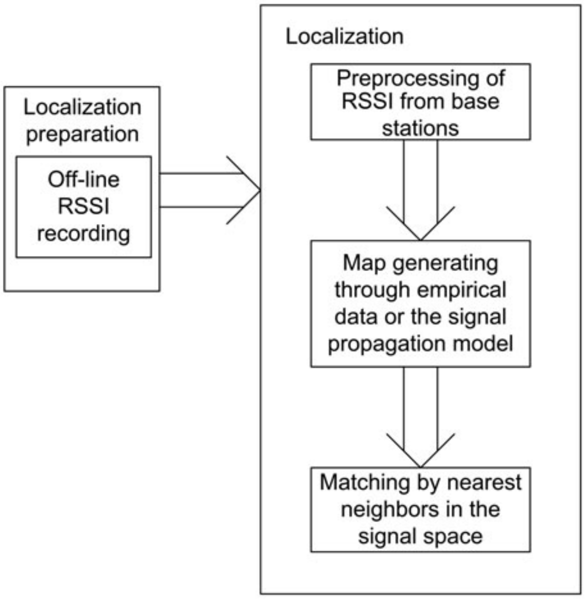

Localization preparation
Off-line RSSI recording
Localization
Preprocessing of RSSI from base stations
Map generating through empirical data or the signal propagation model
Matching by nearest neighbors in the signal space

Fig. 9.15 Two-stage RADAR

the first step, the object's RSSI measurements from the base station are preprocessed for future matching. The sample mean of the multiple measurements to the base station can be adopted to represent the object's location in the signal space. In the second step, an RSSI map of the space is generated from either the empirical data or the propagation model akin to the empirical data. In the final step, the sample mean of the RSSI values from the base station is matched with its nearest neighbor in the signal space.

The major difficulty in implementing RADAR comes from the off-line recording of the RSSI from the base stations. The off-line process of recording RSSI is not cost efficient, because location information needs to be collected together with the RSSI value at pre-determined spots in the indoor space. A kernel-based learning method, aiming at relieving the system from cumbersome off-line preparation, was proposed in [49]. The idea is to formulate the localization problem as a pattern recognition problem with its kernel matrix established on the signal strength matrix, whose entries are the pairwise radio signal strength values collected at sensor nodes. The method requires training data in the learning process. However, the training data can be obtained through automated signal collecting phase involving the anchors and RSSI measurements between pairs of anchors. The pattern recognition algorithm focuses on determining the regions that each node resides in. The centroid of the intersection of the regions, a node belongs to, is thus regarded as the node's location.

Although the localization process of the kernel-based learning method can be executed locally, its training process is inevitably centralized and computation extensive. A fully distributed localization method without explicit statistical model for range measurement was presented in [50]. The location of a node is represented by the exact location and the corresponding uncertaintie. Each node computes its belief of the location, which is a normalized estimate of the posterior likelihood of the location. The node communicates with neighbors on each other's belief and updates its location and the associate belief based on the received information from neighbors. The process iterates until certain convergence criteria are met. The fully distributed algorithm relies on local information and message exchanges, which invokes less communication and computation costs compared to centralized algorithms.

LaSLAT [51] is a framework, based on Bayesian filters to accomplish the task of localizing mobile nodes, in which the location estimates are iteratively updated given batches of new measurements. Extensive empirical studies have shown that LaSLAT can tolerate noisy range measurements and achieve satisfactory location accuracy.

The Kalman filter and particle filter are essentially variants of Bayesian filters, which estimate the state of a dynamic system statistically through noisy measurements. Kalman filters approximate the belief of the state by its first- and second-order moments and achieve the optimality when the initial uncertainty follows Gaussian distribution. In contrast, particle filters realize the Bayesian filter using sets of samples with different importance factors. More discussion on the Bayesian filter and localization can be found in [52].

Statistical approaches are capable of tackling the difficulty of localization introduced by the mobility of nodes. Monte Carlo localization (**MCL**) method was adopted in [53] to solve the localization problem in mobile sensor networks. It follows an approach similar to the Bayesian filter. Unscented Kalman filter, inspired by MCL, which produces location estimates from a subset of samples, was used to localize mobile nodes through passive listening [54].

An alternation to the map-assisted localization is the probabilistic model-based location, in which probabilistic models for range measurements and location estimates are introduced instead of deterministic relationships between range measurements and location estimates. A method, proposed in [55], is based on the probabilistic model of RSSI that is obtained from calibration data corresponding to an outdoor environment without obstructions. The model is given by the following:

$$p \rightarrow lgD \sim N(\mu_D(p), \sigma_D(p)) \tag{9.9}$$

$$\mu_D(p) = lg\widetilde{d} + \sigma_D^2 ln10 \tag{9.10}$$

$$\sigma_D(p) = \sigma_D = \frac{\sigma_P}{10\eta} \tag{9.11}$$

where the distance associated with a particular RSSI value follows a log-normal distribution, σ_P is the variance of the RSSI value, η is the coefficient determined by calibration, and $\widetilde{d}$ is the average of the distances regarding a particular RSSI value. The log-normal model has been verified by the experiment data.

Then the conditional probability density function of the distance can be approximated using (9.12) given RSS measurement with $\overline{P_s}$ and $\sigma_P(s)$.

$$f_D(s|p) = \frac{\xi(s|p)}{\int_0^{\infty} \xi(s|p)ds} \tag{9.12}$$

where

$$\xi(s|p) = \frac{1}{\sqrt{2\pi}\sigma_p(s)} \exp\left(-\frac{(p - \overline{P}(s))^2}{2\sigma_P^2(s)}\right)$$

Initially, the non-anchor node has the estimation of their location to be evenly distributed in the deployment area. After receiving packets from neighboring nodes, either anchor or non-anchor nodes update their estimations on the probability density function (pdf) of the distance. Therefore, the location estimation for the non-anchor nodes can be updated accordingly.

Additionally, a Bayesian model for the noisy distance measurements was reported in [56]. The model is demonstrated in Fig. 9.16.

In the above Bayesian graphical model, conditional density for each vertex is as follows:

X: uniform(0,L), x-ordinate of node, where L is the width of the region.
Y: uniform(0,B), y-ordinate of node, where B is the length of the region.
S_i: $N(b_{i0} + b_{i1} \log Dt_i, \tau_i)$, $i = 1, 2, 3, 4$, RSS, where Dt_i is the distance to the ith access point (anchor node).

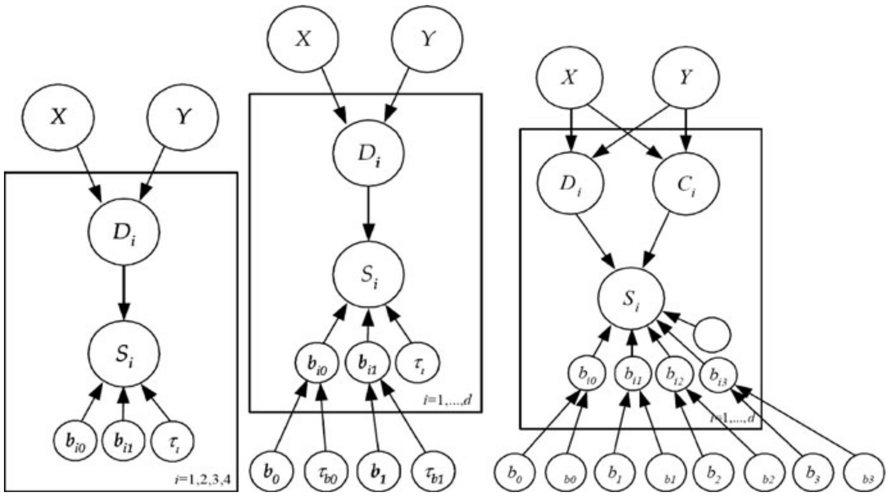


Fig. 9.16 Bayesian graphical models (from *left* to *right*): Bayesian graphical model, hierarchical Bayesian graphical model, and hierarchical Bayesian graphical model with a corridor effect [56]

b_{i0}: N(0,0.001), $i = 1, 2, 3, 4$.
b_{i1}: N(0,0.001), $i = 1, 2, 3, 4$.

where uniform represents uniform distribution and $N(\mu, \tau)$ stands for Gaussian distribution with μ mean and τ standard deviation.

A hierarchical Bayesian graphical model is brought up in order to incorporate the prior knowledge of linear regression models in accordance with the access points that the coefficients of the models are similar to each other. The conditional density for the vertexes is as follows:

X: uniform(0,L), Y: uniform(0,B).
S_i: $N(b_{i0} + b_{i1} \log Dt_i, \tau_i)$, i=1, ..., d.
b_{i0}: $N(b_0, \tau_{b0})$, $i = 1, \ldots, d$, b_0: $N(0, 0.001)$, and τ_{b0}: Gamma(0.001,0.001).
b_{i1}: $N(b_1, \tau_{b1})$, $i = 1, \ldots, d$, b_1: N(0,0.001), and τ_{b1}: Gamma(0.001,0.001).

where Gamma represents the Gamma distribution.

Both of the models are trained with measurement data. A surprising observation on the training result is that the location information of the RSS data does not affect the localization performance obtained from the hierarchical Bayesian graphical model given same amount of sample size. This observation indicates a promising benefit of using hierarchical Bayesian graphical model. When the RSS data are collected, it is not necessary to collect the location associated with the RSS data. It will save a lot of cost on profiling RSS inside a building. Besides the modeling efforts using particular distributions, an empirical study on the statistical characteristics based on probability density functions can be found in [57].

9.8 Summary on Localization Techniques

The focus of our discussion in this section is on the evaluation of localization schemes. The ability to fix the position of a sensor node in terms of absolute location would determine the effectiveness of a particular localization scheme. But, in the absence of GPS or specialized measurement hardware, certain amount of error is bound to creep in. So there is need for qualitative evaluation of the localization schemes.

9.8.1 Localization Accuracy

An extensive body of literature exists on the error analysis that examines the accuracy of estimated locations obtained from various localization schemes. According to the localization process, the sources of localization error may include physical sources, localization algorithms, and refinement process [58, 59].

Table 9.1 Expected accuracy of different measurement technologies[60]

Technology	System	Accuracy	Range
Ultrasound	AHLoS	2 cm	3 m
Ultrawide band	PAL UWB	1.5 m	N/A
RF time of flight	Bluesoft	0.5 m	100 m
Laser time of flight	Laser range finder	1 cm	75 m

The errors due to physical sources are represented by wide range of noises and quantization losses. Ranging techniques vary from ultrasonic to radio, and to laser, etc. A summary on the range accuracy was presented in [60]. Table 9.1 presents a comparison of ranging errors among different range-based techniques.

The most attractive among these are the ones with low-cost and ready-to-use features like time of arrival (TOA) of ultrasonic signal and radio signal strength (RSS) or radio signal strength indication (RSSI). The only concern about these techniques is that they produce highly noisy measurements and are over sensitive to environmental effects.

Localization algorithms encounter two types of error sources. One is system error, which comes from the localization algorithms themselves that work with underlying assumption of accurate range measurement or range-free features. The other source of error is related to connectivity and the fraction of nodes serving as anchors. The last two parameters have significant impacts on the performance of localization algorithms. The effect of system error becomes manageable, when both distance and angle with orientation are available. But the size and the cost of the hardware capable of measuring distance and angle prevent such system from implementation, especially for dense WSNs.

It is of particular interest to study the impact of range errors on the performance of localization algorithms, because range errors are inherent to WSNs employing simple and low-cost range measurement hardware. According to the empirical study on the impact of range errors on multihop localizations [61], high density and Gaussian noises are the two prerequisites for the noisy disk model to work. The study also suggests statistical approaches fix the problem resulted from range errors.

Cramer–Rao lower bound (CRLB) is commonly adopted in the error analysis of the localization schemes. It is a lower bound on the variance of the estimator that estimates the locations. Given the knowledge on the distribution of measurements, it is shown in [62–64] that the bound on the localization error can be obtained through calculating the CRLB. Therefore, the localization schemes are able to evaluate their performances by comparing the localization accuracy with the corresponding CRLB.

9.8.2 Computation and Communication Costs

As energy efficiency is critical to WSNs, it is necessary to consider the computation and communication costs of the localization process in the evaluation of localization schemes. Centralized algorithms like the SDP or MDS-MAP demand range measurements from all the nodes. This is expensive in terms of forwarding the measurements to the processing point and solving the high-dimension matrix. Distributed algorithms, on the other hand, require collaborations among neighboring nodes to some extent. In particular, the multihop localization faces the trade-off between the communication cost on propagating the anchor locations and the degree of accuracy. For the refinement on location estimations, the number of iterations is apparently in the center of the trade-off between the energy consumption for refinement of localization results and the degree of accuracy achievable through refining.

9.8.3 Network and Anchors Density

It is worth noticing that localization algorithms always require a certain level of connectivity. So, localization schemes are based on connectivity, range measurements, angle information, or any combinations thereof. The discussions on the localization algorithms suggest that dense networks lead to better localization performance. However, a dense network does not necessarily guarantee high accuracy in location estimations. The density of the network is usually represented by the number of nodes within an area or the radio range of nodes. The anchor-based localization schemes, aiming at providing absolute locations, require a high density of anchors to ensure low level of localization errors [65].

9.8.4 Summary of Performances

In the previous sections, existing localization schemes were discussed under various scenarios. The difficulty in comparing them is exacerbated by the fact that different test beds for the evaluation purpose are built separately. In the following, let us summarize the performance of these schemes with respect to accuracy, communication/computation costs, and node density.

Table 9.2 presents the simulation results of various localization schemes, where the accuracy was examined through the trade-offs between accuracy and measurement performance, percentage of anchors, deployment of anchors, density of non-anchors, etc. Besides randomly generated networks, a typical deployment of nodes is the grid of non-anchor nodes within a particular area. The localization accuracy of a solution is usually quantified using the average Euclidean distance between the estimated locations and the true locations normalized to the radio range or other system parameters. For mobility-assisted localization, the effect of node density is not as important as in static localization scenarios. In addition, communication/-computation cost may not be of same importance to the off-line simulations as to the real implementations. The table only shows typical values for the items when various trade-offs for one solution were reported in the literature. For the sake of conciseness, radio range and node degree are denoted by R and dg, respectively, in the table, while D represents the average inter-distance of anchors and n is the number of nodes to be localized in the network.

Table 9.2 Summary of simulation results for various localization schemes

Methods	Accuracy	Computation/communication cost	Node density
APIT [9]	40%R	10% message overhead of DV-hop	16 one-hop anchors
Gradient [11]	10%R	N/A	4 anchors at the corners
DV-hop [12]	30%R for isotropic; 90%R for anisotropic	7000 messages exchanged	dg=7.6 with 30% to be anchors
DV-distance [12]	15%R for isotropic; 80%R for anisotropic	7200 messages exchanged	dg=7.6 with 30% to be anchors
Euclidean [12]	10%R for isotropic; 15%R for anisotropic	8000 messages exchanged	dg=7.6 with 30% to be anchors
MDS-MAP [13]	50%R	Computation complexity of O(n^3)	dg$\geqslant$12.2 with 3 anchors at random positions
Ecolocation [66]	30%D	N/A	15 anchors randomly placed
DV-coordinate [27]	1 m for isotropic; 1.25 m for anisotropic	N/A	dg_{average}=9
DV-bearing [26]	1 hop distance	N/A	dg_{average}=10.5
DV-radial [26]	0.8 hop distance	N/A	dg_{average}=10.5
Bisector [47]	5%R	1597 packets	319 randomly deployed nodes
Kernel-based learning [49]	0.47	Worst-case computational time O(n^3)	25 anchors, 400 non-anchors
EKF [67]	20%R	N/A	Randomly deployed nodes, 1 mobile robot
MCL [53]	20%R	50 samples	dg=10, anchor density is 4
RSS model [17]	5%R	O($n^2 log_2 n$)	Node density is 0.5/m^2

Table 9.3 Summary of empirical results

Methods	Accuracy	Computation/ communication cost	Node density
Centroid [6]	1.83m	N/A	4 anchors at corners, grid of non-anchors
Classic MDS [42]	4.3 m for RSS; 1.96 for TOA	O(n^2T) operations	4 anchors at corners
MLE [42]	2.18 m for RSS; 1.23 for TOA	N/A	4 anchors at corners
dwMDS [42]	2.48 m for RSS; 1.12 for TOA	O(nL)	4 anchors at corners
Ecolocation (outdoor) [66]	20%D	N/A	11 nodes with full connectivity
Ecolocation (indoor) [66]	35%D	N/A	12 anchors, 5 non-anchors
Robust quad [34]	5.18 cm for ultrasound ranging	N/A	dg=12, total 40 nodes
Multilateration [21, 45]	10.67 m	N/A	range data from 1 mobile beacon
Mobile beacon [45]	1.4 m	N/A	12 non-anchors, 1 mobile beacon
RADAR [48]	3 m	N/A	dg=3, 3 anchors
Kernel-based learning [49]	3.5 m	N/A	Grid deployment of 25 anchors and 81 nodes
LaSLAT [51]	1.9 cm for ultrasound ranging	N/A	dg=10, total 27 nodes

Some of the empirical results of localization schemes are listed in Table 9.3. The number of anchors is usually low due to the cost and the difficulty of deployment. These results echo the report on the high accuracy achieved by ultrasound ranging techniques. The analysis on the communication/computation costs was not extensively presented in the empirical studies due to the difficulty in measuring the cost in real implementations.

9.9 Open Issues

There has been extensive research on sensor localization; however, there are some important open issues specially relevant to sensor nodes in a WSN which either remain unresolved or not explored extensively. Some of these issues are listed below.

1. Energy consumption
   Although energy consumption has been addressed in the study on localization with WSNs, the energy efficiency goal of the localization schemes remains challenging. The problem of minimizing energy consumption of the localization process deserves further attention. As the energy consumption of the localization application involves measurements, communication with neighbors, and estimation of locations, the task of quantifying the energy consumption demands system-wide efforts to incorporating energy-efficient design at all communication layers and all aspects of the localization algorithm.
2. Three-dimensional localization
   The typical scenario for localization with WSNs is to find out locations of the nodes in a 2-D plane. However, nodes are usually deployed in a 3-D space, which leads to differences on both ranging results and localization algorithms. Analysis on localization schemes focusing on the 3-D space is of particular interests to real applications of WSNs, especially when the difference between 2-D space and 3-D space is significant. For instance, irregularity of the radio transmission has been investigated in 2-D space, while its counterpart in 3-D space remains to be unknown [68, 69].
3. Security and privacy
   Security and privacy have always been the fundamental issues in large-scale deployments of WSNs. Since locations of nodes are of importance to the applications' tasks, the security of the location needs to be guaranteed. Although some researches on security of localization schemes are presented [70, 71], the types of attacks and the related countermeasures are restricted to a few typical cases. Similarly, researches on privacy of nodes' locations mostly focus on preventing the locations of data sources or base stations from being exposed to adversaries [72, 73]. The existing sensor localization schemes have not been fully examined from the perspective of privacy protection.

9.10 Conclusions

In this chapter, we discussed sensor localization from its algorithmic aspects. Graph-theoretic techniques solve the localization problem through modeling the proximity information collected by the sensor nodes into edges of the graph and the nodes themselves vertices. In order to reduce the estimation error on nodes' locations, geometric-based techniques were proposed to incorporate the distance measurements between pairwise nodes in the localization process. Furthermore, techniques with additional hardware explore the geometric characteristic of the network by introducing the angle measurements. Besides, iterative techniques focus on refining the initial estimation of the locations until the goal of high estimation accuracy is achieved. In contrast to localization of static sensor nodes, mobile-assisted localization techniques exploit the presence of mobile nodes that have abundant resources. The mobile nodes are able to tackle the problem of low density of anchors in the network by moving around static sensor nodes acting as the anchor. However, mobility poses additional difficulty to the sensor localization problem when all the sensor nodes are mobile. Statistical techniques are able to accomplish the localization task for mobile nodes with the help of statistical models for range measurements and the learning mechanism. The comparison among the existing algorithms for sensor localization shows that energy efficiency of the localization process remains to be a critical issue for wireless sensor networks, while security and privacy in sensor localization among other open issues expect further investigations.

Acknowledgments This work is partially supported by an EADS grant and NSF grants CNS-0916221 and CNS-0721951 and a Research I grant.

References

1. S. Capkun, M. Hamdi, and J.-P. Hubaux. GPS-free positioning in mobile ad-hoc networks. *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, page 9008, 2001.
2. G. Mao and B. Fidan. Localization algorithms and strategies for wireless sensor networks. *Information Science Reference – Imprint of: IGI Publishing*, pages 1–526, 2009.
3. A. Savvides, M. Srivastava, L. Girod, and D. Estrin. Localization in sensor networks. In C. S. Raghavendra, K. M. Sivalingam, and T. Znati, editors, *Wireless Sensor Networks*, Kluwer, Norwell, MA, pages 327–349, 2004.
4. J. Bachrach and C. Taylor. Localization in sensor networks. In I. Stojmenovic editor *Handbook of Sensor Networks: Algorithms and Architectures*, Wiley, New York, NY, ch. 9, pages 277–310, 2005.
5. G. Mao, B. Fidan, and B. D. O. Anderson. Wireless sensor network localization techniques. *Computer Network*, 51(10):2529–2553, 2007.
6. N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low-cost outdoor localization for very small devices. *IEEE Personal Communications Magazine [see also IEEE Wireless Commun. Mag.]*, 7(5):28–34, 2000.
7. P. Agrawal, R. K. Ghosh, and S. K. Das. Localization of wireless sensor nodes using proximity information. In *ICCCN 2007: The 16th International Conference on Computer Communications and Networks*, pages 485–490, 2007.

8. R. Salomon. Precise localization in coarse-grained localization algorithms through local learning. In *IEEE SECON 2005: Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pages 533–540, 2005.
9. T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher. Range-free localization and its impact on large scale sensor networks, *Transactions on Embedded Computing Systems*, 4(4):877–906, 2005.
10. H.-C. Chu and R.-H. Jan. A GPS-less, outdoor, self-positioning method for wireless sensor networks. *Ad Hoc Networks*, 5(5):547–557, 2007.
11. R. Nagpal, H. Shrobe, and J. Bachrach. Organizing a global coordinate system from local information on an ad hoc sensor network. In *IPSN 2003: Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, 2003.
12. D. Niculescu and B. Nath. Ad hoc positioning system (APS). *IEEE GLOBECOM 2001*, pages 2926–2931, 2001.
13. Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz. Localization from mere connectivity. In *MobiHoc 2003: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 201–212, 2003.
14. K. Whitehouse, C. Karlof, and D. Culler. A practical evaluation of radio signal strength for ranging-based localization. *SIGMOBILE Mobile Computing and Communications Review*, 11(1):41–52, 2007.
15. R. Reghelin and A. A. Fröhlich. A decentralized location system for sensor networks using cooperative calibration and heuristics. In *MSWiM 2006: Proceedings of the 9th ACM International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 139–146, 2006.
16. H. Lim, L.-C. Kung, J. C. Hou, and H. Luo. Zero-configuration, robust indoor localization: Theory and experimentation. *INFOCOM 2006: 25th IEEE International Conference on Computer Communications*, pages 1–12, April 2006.
17. R. Peng and M. L. Sichitiu. Probabilistic localization for outdoor wireless sensor networks. *SIGMOBILE Mobile Computing and Communications Review*, 11(1):53–64, 2007.
18. X. Cheng, A. Thaeler, G. Xue, and D. Chen. TPS: a time-based positioning scheme for outdoor wireless sensor networks. *INFOCOM 2004: 23rd AnnualJoint Conference of the IEEE Computer and Communications Societies*, 4:2685–2696, 2004.
19. J. Zhang, T. Yan, J. A. Stankovi, and S. H. Son. Thunder: towards practical, zero cost acoustic localization for outdoor wireless sensor networks. *SIGMOBILE Mobile Computing and Communications Review*, 11(1):15–28, 2007.
20. M. Broxton, J. Lifton, and J. A. Paradiso. Localization on the pushpin computing sensor network using spectral graph drawing and mesh relaxation. *SIGMOBILE Mobile Computing and Communications Review*, 10(1):1–12, 2006.
21. A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *MobiCom 2001: the 7th Annual International Conference on Mobile Computing and Networking*, pages 166–179, 2001.
22. M. Youssef, A. Youssef, C. Rieger, U. Shankar, and A. Agrawala. PinPoint: An asynchronous time-based location determination system. In *MobiSys 2006: the 4th International Conference on Mobile Systems, Applications and Services*, pages 165–176, 2006.
23. T. Eren, O. Goldenberg, W. Whiteley, Y. Yang, A. Morse, B. Anderson, and P. Belhumeur. Rigidity, computation, and randomization in network localization. *INFOCOM 2004: The 23rd AnnualJoint Conference of the IEEE Computer and Communications Societies*, 4: pages 2673–2684, 2004.
24. J. Aspnes, D. Goldenberg, and Y. R. Yang. On the computational complexity of sensor network localization. In *The 1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, pages 32–44, 2004.
25. A. M.-C. So and Y. Ye. Theory of semidefinite programming for sensor network localization. *Mathmatical Programming*, 109(2):367–384, 2007.

26. D. Niculescu and B. Nath. Ad hoc positioning system (APS) using AOA. *INFOCOM 2003: The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, 3:1734–1743, 2003.
27. D. Niculescu and B. Nath. Dv based positioning in ad hoc networks. *Telecommunication Systems*, 22(1–4):267–280, 2003.
28. Y. Zhang and L. Cheng. PLACE: protocol for location and coordinate estimation: A wireless sensor network approach. *Computing Network*, 46(5):679–693, 2004.
29. K. Chintalapudi, A. Dhariwal, R. Govindan, and G. Sukhatme. Ad-hoc localization using ranging and sectoring. *INFOCOM 2004: The 23rd AnnualJoint Conference of the IEEE Computer and Communications Societies*, 4:2662–2672, 2004.
30. A. Savvides, H. Park, and M. B. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. In *WSNA 2002: The 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 112–121, ACM, New York, NY, 2002.
31. L. Doherty, K. Pister, and L. E. Ghaoui. Convex position estimation in wireless sensor networks. *INFOCOM 2001: The 20th Annual Joint Conference of the IEEE Computer and Communications Societies*, 3:1655–1663, 2001.
32. P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Transaction on Sensor Networks*, 2(2):188–220, 2006.
33. X. Ji and H. Zha. Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling. *INFOCOM 2004: The 23rd AnnualJoint Conference of the IEEE Computer and Communications Societies*, 4: 2652–2661, 2004.
34. D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. *SenSys 2004: The 2nd International Conference on Embedded Networked Sensor Systems*, pages 50–61, 2004.
35. P. Stoica and K. Sharman. Maximum likelihood methods for direction-of-arrival estimation, *IEEE Transactions on Acoustics Speech and Signal Processing*, 38(7):1132–1143, 1990.
36. N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system, *MobiCom 2000: The 6th Annual ACM International Conference on Mobile Computing and Networking*, 2000.
37. J. Bruck, J. Gao, and A. A. Jiang. Localization and routing in sensor networks by local angle information. *MobiHoc 2005: The 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 181–192, 2005.
38. M. Bǎdoiu, E. D. Demaine, M. T. Hajiaghayi, and P. Indyk. Low-dimensional embedding with extra information. *SCG 2004: The 20th Annual Symposium on Computational Geometry*, pages 320–329, 2004.
39. O.-H. Kwon and H.-J. Song. Localization through map stitching in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(1):93–105, 2008.
40. R. Iyengar and B. Sikdar. Scalable and distributed GPS free positioning for sensor networks. *ICC 2003: IEEE International Conference on Communications*, 1:338–342, 2003.
41. H. Chan, M. Luk, and A. Perrig. Using clustering information for sensor network localization. *DCOSS 2005: IEEE Conference on Distributed Computing in Sensor Systems*, 2005.
42. J. A. Costa, N. Patwari, and A. O. H. Iii. Distributed weighted-multidimensional scaling for node localization in sensor networks, *ACM Transactions Sensor Networks*, 2:39–64, 2006.
43. J. Liu, Y. Zhang, and F. Zhao, Robust distributed node localization with error management. *MobiHoc 2006: The 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 250–261, 2006.
44. S. Nikoletseas and P. Spirakis. Efficient sensor network design for continuous monitoring of moving objects. *Theoretical Computer Science*, 402(1):56–66, 2008.
45. M. L. Sichitiu and V. Ramadurai. Localization of wireless sensor networks with a mobile beacon. *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 174–183, 2004.

46. A. Galstyan, B. Krishnamachari, K. Lerman, and S. Pattem. Distributed online localization in sensor networks using a moving target. *IPSN 2004: The 3rd International Symposium on Information Processing in Sensor Networks*, pages 61–70, 2004.
47. K.-F. Ssu, C.-H. Ou, and H. Jiau. Localization with mobile anchor points in wireless sensor networks, *IEEE Transactions on Vehicular Technology*, 54(3):1187–1197, 2005.
48. P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. *INFOCOM 2000: The 19th IEEE International Conference on Computer Communications*, pages 775–784, 2000.
49. X. Nguyen, M. I. Jordan, and B. Sinopoli. A kernel-based learning approach to ad hoc sensor network localization. *ACM Transactions Sensor Networks*, 1(1):134–152, 2005.
50. E. T. Ihler, J. W. Fisher, O. L. Moses, and A. S. Willsky. Nonparametric belief propagation for self-localization of sensor networks. *IEEE Journal on Selected Areas Communications*, 23:809–819, 2005.
51. C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue. Simultaneous localization, calibration, and tracking in an ad hoc sensor network. *IPSN 2006: The 5th International Conference on Information Processing in Sensor Networks*, pages 27–33, 2006.
52. D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. Bayesian filtering for location estimation. *IEEE Pervasive Computing*, 2(3):24–33, 2003.
53. L. Hu and D. Evans. Localization for mobile sensor networks. *MobiCom 2004: The 10th Annual International Conference on Mobile Computing and Networking*, pages 45–57, 2004.
54. L. Zhang, Q. Cheng, Y. Wang, and S. Zeadally. Landscape: a high performance distributed positioning scheme for outdoor sensor networks. *WiMob 2005: IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, 3:430–437, 2005.
55. R. Peng and M. L. Sichitiu. Probabilistic localization for outdoor wireless sensor networks. *SIGMOBILE Mobile Computing Communication Review*, 11(1):53–64, 2007.
56. D. Madigan, E. Einahrawy, R. P. Martin, W. H. Ju, P. Krishnan, and A. S. Krishnakumar. Bayesian indoor positioning systems. *INFOCOM 2005: The 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2:1217–1227, 2005.
57. A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin. Statistical model of lossy links in wireless sensor networks. *IPSN 2005: The 4th International Symposium on Information Processing in Sensor Networks*, 2005.
58. S. Slijepcevic, S. Megerian, and M. Potkonjak. Location errors in wireless embedded sensor networks: sources, models, and effects on applications. *SIGMOBILE Mobile Computing Communications Review*, 6(3):67–78, 2002.
59. H. A. Oliveira, E. F. Nakamura, A. A. F. Loureiro, and A. Boukerche. Error analysis of localization systems for sensor networks, *GIS 2005: The 13th Annual ACM International Workshop on Geographic Information Systems*, pages 71–78, 2005.
60. A. Savvides, W. Garber, R. Moses, and M. Srivastava. An analysis of error inducing parameters in multihop sensor node localization. *IEEE Transactions Mobile Computing*, 4(6):567–577, Nov.-Dec. 2005.
61. K. Whitehouse, C. Karlof, A. Woo, F. Jiang, and D. Culler. The effects of ranging noise on multihop localization: an empirical study. *IPSN 2005: The 4th International Symposium on Information Processing in Sensor Networks*, pages 73–80, 2005.
62. N. Patwari, I. A.O. Hero, M. Perkins, N. Correal, and R. O'Dea. Relative location estimation in wireless sensor networks. *IEEE Trans. Signal Processing* , 51(8):2137–2148, 2003.
63. C. Chang and A. Sahai. Estimation bounds for localization. *SECON 2004: The 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pages 415–424, 2004.
64. S. Venkatesh and R. M. Buehrer. Multiple-access insights from bounds on sensor localization. *Pervasive Mobile Computing*, 4(1): 33–61, 2008.
65. N. Bulusu, J. Heidemann, D. Estrin, and T. Tran. Self-configuring localization systems: Design and experimental evaluation. *Transactions on Embedded Computing Systems*, 3(1):24–60, 2004.

66. K. Yedavalli, B. Krishnamachari, S. Ravula, and B. Srinivasan. Ecolocation: A sequence based technique for RF localization in wireless sensor networks. *IPSN 2005: The 4th International Symposium on Information Processing in Sensor Networks*, pages 285–292, 2005.
67. K. Sreenath, F. L. Lewis, and D. O. Popa. Simultaneous adaptive localization of a wireless sensor network. *SIGMOBILE Mobile Computing Communication Review*, 11(2):14–28, 2007.
68. A. Y. Teymorian, W. Cheng, L. Ma, X. Cheng, X. Lu, and Z. Lu. 3d underwater sensor network localization. *IEEE Transactions. Mobile Computing.*, 8:1610–1621, 2009.
69. H. S. Abdelsalam and S. Olariu. A 3d-localization and terrain modeling technique for wireless sensor networks. In *FOWANC 2009: The 2nd ACM International Workshop on Foundations of Wireless Ad Hoc and Sensor Networking and Computing*. ACM, New York, NY, pages 37–46, 2009.
70. Z. Li, W. Trappe, Y. Zhang, and B. Nath. Robust statistical methods for securing wireless localization in sensor networks. *IPSN 2005: The 4th International Symposium on Information Processing in Sensor Networks*, pages 91–98, 2005.
71. L. Lazos, R. Poovendran, and S. Čapkun. ROPE: robust position estimation in wireless sensor networks. *IPSN 2005: The 4th International Symposium on Information Processing in Sensor Networks*, 2005.
72. M. Shao, Y. Yang, S. Zhu, and G. Cao. Towards statistically strong source anonymity for sensor networks. In *INFOCOM 2008: The 27th IEEE International Conference on Computer Communications*, pages 51–55, 2008.
73. Y. Jian, S. Chen, Z. Zhang, and L. Zhang. Protecting receiver-location privacy in wireless sensor networks. In *INFOCOM 2007: The 26th IEEE International Conference on Computer Communications*, pages 1955–1963, 2007.

Chapter 10
Spatio-temporal Context in Wireless Sensor Networks

Anahit Martirosyan and Azzedine Boukerche

Abstract Context represents any knowledge obtained from Wireless Sensor Networks (WSNs) regarding the object being monitored. Context-awareness is an important feature of WSN applications as it provides an ultimate tool for making the applications "smart". The information about a sensed in a WSN phenomenon is comprehensive only when it includes the geographical location and the time of occurrence of the phenomenon. Thus, the location and time are essential constituents of WSNs' context, though the concept of context is not limited to only space and time. In this chapter, we consider the spatio-temporal context of WSNs as it serves as a foundation for context-aware systems. In order to build context-aware WSNs it is necessary to consider three areas concerning the spatio-temporal correlation of events sensed in WSNs: node localization, temporal event ordering and time synchronization. While localization's task is to provide geographic coordinates of a sensed event, preserving temporal relationships of the events in WSNs is necessary for ensuring their correct interpretation at the monitoring centre and for taking proper and prompt actions. The latter can be achieved by guaranteeing time synchronization and temporal event ordering mechanisms. We present an overview of selected algorithms in each of the areas, first providing the necessary background and then presenting a comparison of features of the discussed algorithms.

10.1 Introduction

Context represents any knowledge obtained from a Wireless Sensor Network (WSN) such as the time and location of a sensed event as well as other knowledge about the object being monitored. Context-awareness is an important feature of WSN applications as it provides an ultimate tool for making the applications "smart". The objective of context-aware applications is to use this knowledge in order to adapt the application to the specifics of the object being monitored. The knowledge about a sensed phenomenon is comprehensive only when it is accompanied with the

A. Martirosyan(✉)
University of Ottawa, 800 King Edward Ave, Ottawa, ON Canada
e-mail: amart013@uottawa.ca

geographical location of the node that sensed the phenomenon as well as the time of the event's occurrence. Thus, the location and time of a sensed phenomenon in a WSN are important constituents of context. The task of node localization algorithms is to provide geographic coordinates of sensor nodes. Temporal relationships of the events in WSNs are ensured by means of time synchronization algorithms and temporal event ordering algorithms. The concept of context is discussed in the following section in more detail.

10.1.1 What Is Context?

Even though most people tacitly understand what context is, it is somewhat difficult to explain. The term context-aware system was first defined by Schilit et al. [41] as software that adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time. We adopted the definition of context given by Dey [10], "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves". With an understanding of what context is, application designers can determine what behaviours or features the applications should support.

Examples of context-aware computing applications are Smart Homes and Ubiquitous Health Monitoring Systems, in which the environment and user are initially monitored (generally by means of wireless sensor networking technology) to collect information about the conditions and preferences of the object being monitored. This knowledge (context) is then used to adapt the application to the user's needs. The spatio-temporal context provides a foundation for context-aware applications. Once the context about location and space is considered, then the context can be extended to include other knowledge about the object being monitored. For instance, in the case of an Ubiquitous Health Monitoring System, the information about a patient that is being monitored (such as heartbeat, blood pressure, temperature etc.) will be collected and analyzed by the Sink(s) (such as a physician, hospital etc.). The knowledge about the patient being monitored will then be included into the context about the patient. By taking into account the context, the health monitoring system will thus be adapted to the specific patient: observe his/her conditions and predict the evolution of events with a certain probability.

The remainder of the chapter is organized as follows. We present an overview of selected algorithms in the areas of node localization, temporal event ordering and time synchronization in WSNs in Sects. 10.2, 10.3, and 10.4, respectively. Section 10.5 concludes the chapter.

10.2 Node Localization in WSNs

The problem of node localization in WSNs has received an increased attention in the research community. A survey on localization algorithms can be found in the references [3, 19]. There have been proposed a number of approaches to node localization in static WSNs [1, 9, 38, 42, 46], to name just a few. Other approaches presented in the references [4, 24, 32, 44, 47, 49] considered mobile node-assisted localization.

10.2.1 The Task of Localization Algorithms for WSNs

Due to the ad hoc nature of the WSNs and the limited capabilities of sensor nodes, localization faces the challenges of providing accuracy while minimizing computational cost and preserving network's scarce energy resources. A small percentage of sensor nodes may be equipped with either global or local coordinates (these nodes are called beacons or anchors), while the rest of the nodes are location unaware (these nodes are called unknowns). The goal of localization protocols in WSNs is to provide location information (geographic coordinates) for as many sensor nodes as possible. In order for a sensor node to be able to locate itself in a two/three-dimensional space it is necessary for the node to know the positions of three/four nodes as well as the distances to these nodes (the ranging techniques for obtaining distances are overviewed in Sect. 10.2.2). After a sensor node obtains the required information regarding the beacons, it is able to calculate its own geographic coordinates by performing trilateration. In the case when the angles to the beacons are known instead of the distances, the sensor node will perform triangulation instead in order to obtain its position. It is not always possible for a node to receive the information about the beacon nodes in one hop. In these cases, a multi-hop propagation of the beacon information is used for the sensor nodes to learn of the positions of beacons and the corresponding distances/angles to them.

In the following subsection we first briefly overview the ranging techniques for distance/angle estimation (a detailed description of the techniques can be found in the reference [3]). We then overview trilateration and multilateration techniques for obtaining sensor nodes' geographic coordinates.

10.2.2 Estimation of Distances and Angles

- *Received Signal Strength Indication* (RSSI) stems from the fact that radio signal strength is inversely proportional to the distance squared. Thus, if a receiving sensor node measures the received signal strength of a sending node, it should be able to obtain the distance between the sending node and itself. RSSI is practical in the sense of low cost as no additional equipment is required in order to measure the distance between two nodes. Every sensor node is equipped with a radio module and thus is able to calculate the distance. However, RSSI is inaccurate as the measurements are noisy on the order of several meters [2]. The inaccuracy of RSSI can be explained by the fact that radio signal propagation is not universal for different materials.

- *Time of Arrival* (ToA) estimates the distance between the sender and receiver sensor nodes by using the time that it takes for the signal to propagate from one node to another. The radio signal propagates with the speed of light and if the time that it took the signal to propagate from the sender to the receiver is known, then the receiver node can compute the distance separating the two nodes. Obviously, the nodes need to be synchronized in order for the computation to be accurate.
- *Angle of Arrival* (AoA) is obtained by means of directional antennas or using a set of receiver nodes. By using the time of arrival of the signal at the receivers it is possible to calculate the angle of arrival at the sensor node in question. This technique has an accuracy of a few degrees. However, it requires additional hardware and thus adds to the cost and size of a sensor node.
- *Time Difference of Arrival* (TDoA) uses two different kinds of signals to compute the distance between the sending and receiving nodes. Radio signal can be sent at the same time (or with a delay) with an acoustic signal [48], for instance. The difference in the speed of the two signals allows the receiver node to calculate the distance that the signals have traveled by multiplying the difference in the speeds by the difference in the time of arrival of the two signals. This ranging technique has a very high precision, on the order of centimeters. The main disadvantage of the technique is that it requires extra hardware, which increases the cost and size of a sensor node.
- *Hop Count* cannot be categorized as a ranging technique; however the metric is used by many localization algorithms for WSNs to obtain distance measurements to the beacon nodes. The DV-hop-based algorithm [30] is the pioneering approach in this category of algorithms. The main idea is based on the fact that if messages sent by two nodes are able to reach each other, then the distance between the nodes is bounded by the transmission range of their radios. The hop count between two nodes is defined as the number of hops on the shortest path connecting the two nodes. The advantage of the hop count-based approaches is their simplicity and low cost as they do not require any additional hardware and simply use the connectivity information to obtain distance estimates between the nodes. However, the inaccuracy of the approach is high due to the accumulated error that results from the averaging of the hop distance between two neighboring nodes. The inaccuracies may be high in the cases when two nodes are separated by an obstacle.

10.2.3 Trilateration

Once a node has received the information about three beacons and the corresponding distances to them, it performs trilateration and obtains its position estimate. In order to be able to perform trilateration, the unknown node U depicted in Fig. 10.1 needs to know the coordinates of the beacon nodes B_1, B_2 and B_3 as well as the distances to these beacon nodes (r_1, r_2, r_3). Using the theorem of Pythagoras the information about the beacons' coordinates and the distances to the beacon nodes from the unknown node U can be expressed in Eqs. (10.1), (10.2), and (10.3).

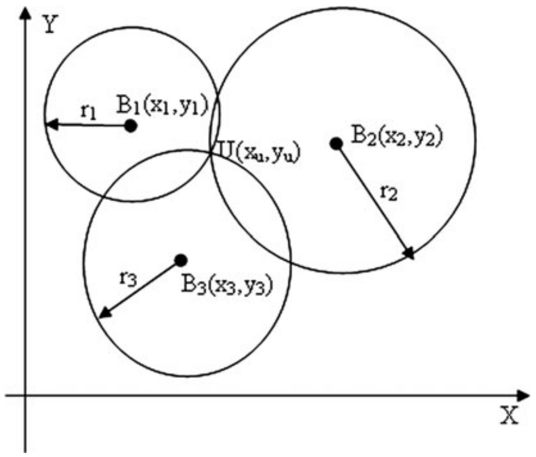


Fig. 10.1 Trilateration to the three beacon nodes B$_1$, B$_2$ and B$_3$ is performed by the node U to find its own coordinates (x$_u$,y$_u$)

$$(x_1 - x_u)^2 + (y_1 - y_u)^2 = r_1^2 \tag{10.1}$$

$$(x_2 - x_u)^2 + (y_2 - y_u)^2 = r_2^2 \tag{10.2}$$

$$(x_3 - x_u)^2 + (y_3 - y_u)^2 = r_3^2 \tag{10.3}$$

Subtracting Eq. (10.3) from Eqs. (10.1) and (10.2) will give

$$(x_1 - x_u)^2 - (x_3 - x_u)^2 + (y_1 - y_u)^2 - (y_3 - y_u)^2 = r_1^2 - r_3^2$$
$$(x_2 - x_u)^2 - (x_3 - x_u)^2 + (y_2 - y_u)^2 - (y_3 - y_u)^2 = r_2^2 - r_3^2$$

After rearranging the equations, we get a system of two linear equations with two unknowns as shown below. The coordinates of the unknown node U(x$_u$, y$_u$) can be obtained by solving the following system of equations.

$$2(x_3 - x_1)x_u + 2(y_3 - y_1)y_u = \left(r_1^2 - r_3^2\right) - \left(x_1^2 - x_3^2\right) - \left(y_1^2 - y_3^2\right)$$
$$2(x_3 - x_2)x_u + 2(y_3 - y_2)y_u = \left(r_2^2 - r_3^2\right) - \left(x_2^2 - x_3^2\right) - \left(y_2^2 - y_3^2\right)$$

When the information about angles to the beacon nodes is used instead of the distances, *triangulation* is performed by an unknown node by means of using the laws of sines and cosines to compute its coordinates.

10.2.4 Multilateration

When more than three beacon nodes are available, an unknown node can use multilateration to obtain its geographic coordinates. Multilateration may yield a more accurate position than trilateration as the information to more beacon nodes is taken into account by an unknown node to calculate its coordinates. The multilateration equations are shown in Eqs. (10.4), (10.5), and (10.6), when there are n beacon nodes and the distance errors e are considered.

$$(x_1 - x_u)^2 + (y_1 - y_u)^2 = r_1^2 - e \tag{10.4}$$

$$(x_2 - x_u)^2 + (y_2 - y_u)^2 = r_2^2 - e \tag{10.5}$$

$$\vdots$$

$$(x_3 - x_u)^2 + (y_3 - y_u)^2 = r_3^2 - e \tag{10.6}$$

Similar to the case of trilateration, the system of linear equations can be rewritten in a matrix form Ax $\approx$ b by subtracting the last equation as shown in Eq. (10.7). The system of equations is over-determined as there are more equations than unknowns. It can be solved by using the least squares method [16, 21, 38]. Then, x = (A^{T}A)$^{-1}$(A^{T}b), where A^T is a transpose of the matrix A. The least squares method minimizes the sum of the squares of the differences between estimated and computed distances.

$$2\begin{bmatrix} x_n - x_1 & y_n - y_1 \\ \dots & \dots \\ x_n - x_{n-1} & y_n - y_{n-1} \end{bmatrix} \begin{bmatrix} x_u \\ y_u \end{bmatrix} \approx \begin{bmatrix} \left(r_1^2 - r_n^2\right) - \left(x_1^2 - x_n^2\right) - \left(y_1^2 - y_n^2\right) \\ \dots \\ \left(r_{n-1}^2 - r_n^2\right) - \left(x_{n-1}^2 - x_n^2\right) - \left(y_{n-1}^2 - y_n^2\right) \end{bmatrix} \tag{10.7}$$

Another method for calculating unknown node's coordinates is *bounding box* [3, 39], in which squares around the beacon nodes are used (instead of circles as in lateration) to represent the boundaries of each beacon node. The unknown node is then located at the intersection of the squares.

10.2.5 Localization Algorithms for WSNs

The *Global Positioning System* (GPS) [31] is the most known and used location system. In GPS, the ranges to at least four from the existing 24 satellites are used for lateration in order to find the coordinates of the receiver that needs to be localized. However, GPS is not suitable to be used in WSNs for a number of reasons. From a point of view of the low price and small size of a sensor node, it is not reasonable to equip every sensor node with a fairly expensive and large in size GPS unit. Next, GPS's energy consumption is high and thus its use is prohibitive in energy-constrained WSNs. Also, GPS cannot be employed indoors as it requires a line of sight.

10.2.5.1 Ad Hoc Positioning System

One of the pioneering approaches to node localization for WSNs is the *Ad Hoc Positioning System* (APS) [30], which has three different versions as described below. The APS is similar to distance vector routing, in which every node communicates only with its immediate neighbours. Each message exchanged in the APS contains the node's available estimates to landmarks (beacons). The method uses multi-hop propagation from landmarks to sensor nodes. Once a node receives three range estimates to three or more landmarks, it is able to compute its own position by using trilateration.

In the *DV-hop* propagation method, there are three non-overlapping stages. In the first stage, each node propagates distances in hops to each landmark in the WSN. In the second phase, a landmark estimates an average size of a hop after it receives a message from another landmark. It then floods the networks with the average size of a hop in meters as a correction to distance measurements. In the third phase, an arbitrary node, after receiving the correction, estimates distances to a landmark in meters and uses this information to perform trilateration. Another version of the algorithm is *DV-distance*, in which, as opposed to *DV-hop*, the distances between neighbouring nodes are measured using radio signal strength RSSI and are propagated in meters rather than hops. *Euclidean* propagation method propagates true *Euclidean* distances to landmarks. The advantage of *DV-hop* method is its simplicity. The fact that no range measurements are used by an arbitrary node to perform trilateration frees the approach from the inaccuracies inherent to range measurement techniques RSSI, AoA, ToA and TDoA [3]. However, the approach is not scalable mainly due to the high communication cost.

10.2.5.2 Robust Positioning Algorithms for Distributed Ad Hoc Wireless Sensor Networks

Another DV-hop-based algorithm is *Robust Positioning Algorithms for Distributed Ad Hoc Wireless Sensor Networks* (RPA) [37]. The algorithm goes through two phases, namely Hop-TERRAIN and Refinement. In the Hop-TERRAIN phase, the algorithm finds the number of hops from a node to each beacon node in the network. The nodes then multiply the hop count by a shared metric of an average hop distance to estimate the distance to each of the anchor nodes. Triangulation is performed by using the least squares algorithm. In the refinement phase, given the position estimates of the Hop-TERRAIN, the goal is to obtain more accurate positions by means of using ranges between neighbouring nodes. Refinement goes through iterations, in which each node broadcasts its position estimate and waits for replies with position estimates from its one-hop neighbours. By using one of the above-mentioned ranging techniques, the node also obtains distance measurements to its neighbouring nodes. It then computes a least squares lateration to obtain its new position. As a result, in the majority of cases, after a number of iterations, the node's position will change towards its true position. To mitigate error propagation, the refinement algorithm assigns a confidence level to each node's position. The confidence levels are used to weigh the linear equations when solving the system of linear equations by means of the least squares trilateration. The anchors have high confidence levels, whereas a node that observes poor conditions (e.g. few neighbours, poor constellation) associates a low confidence level with its position estimate. Consequently, the nodes with poor confidence levels about their position estimates have less impact on the outcome of the triangulations performed by the neighbouring nodes.

10.2.5.3 Ad Hoc Localization System

The *Ad Hoc Localization System* (AHLoS) [40] goes through a two-phase process—ranging and estimation—to dynamically discover nodes' locations. During the ranging phase, each node estimates its distances from neighbouring nodes. In the estimation phase, nodes use the ranging information and the known beacon locations to estimate their positions. Once a node has a position estimate, it becomes a beacon and assists other nodes in estimating their positions in an iterative manner. RSSI and ToA are employed to obtain ranging information in AHLoS.

Two versions of the algorithm are considered, such as centralized and distributed. In the centralized version of the algorithm, location estimation is performed at a central, more powerful node. After that the results are forwarded to the nodes. In the algorithm's distributed version, the location estimation is performed at the nodes, without involving a central node. The algorithm uses multilateration when a node has ranging information to more than three beacons. Atomic multilateration, iterative multilateration and collaborative multilateration algorithms are used in the AHLoS. In the atomic multilateration, if a node has the ranging information to more than three beacon nodes, it performs multilateration. In the case of iterative multilateration, such a node then becomes a beacon after it estimates its position. It then assists the neighbouring nodes in obtaining position estimates. The collaborative multilateration is considered when two neighbouring nodes do not have ranging information to the required number of beacons, but may assist each other (collaborate) in obtaining the needed information. The iterative multilateration can be applied in small-scale networks.

10.2.5.4 The *n*-Hop Multilateration Primitive for Node Localization Problems

The *n-Hop Multilateration* approach [39], as its title suggests, considers a multilateration that spans over multiple hops. The algorithm goes through three main phases and a post-processing phase. In the first phase, the nodes self-organize into groups, collaborative subtrees, so that the nodes that do not have position estimates are over-constrained and can have only one possible solution.

If a node does not satisfy the required constraints, it does not become part of a collaborative subtree. During the second phase, the nodes use geometric relationships (a method of constructing a bounding box [3, 39]) between measured distances and beacon locations to obtain initial position estimates. In the third phase—refinement—iterative least squares approach is used to obtain final position estimates. The position estimates are further refined in the post-processing phase of the algorithm similarly to the process used in the second phase. The n-hop collaborative multilateration approach to node localization is scalable to larger networks as opposed to multilateration approach described above [40].

10.2.5.5 A Lightweight Iterative Positioning Algorithm for Wireless Sensor Networks

The *Lightweight Iterative Positioning* (LIP) [28] algorithm for WSNs is a DV-hop-based algorithm, in which localization is achieved in two phases: initial position estimation and iterative refinement. The main objective of the proposed solution is to reduce the communication overhead of the DV-hop-based localization algorithms without affecting the accuracy. The algorithm stems from the observation that the synchrony in the nodes' localization process in the DV-hop-based algorithms may increase communication cost and delay of node localization, in addition to occurrence of collisions and lost messages. The randomization of node localization is achieved in a number of ways as discussed below. In the initial position estimation phase, the algorithm attempts to decrease communication overhead by randomizing the process of propagation of beacon position information to the nodes in the network by using a random Time-To-Live (TTL) (number of hops the message propagates) for the most of the messages with beacons' position information.

The method that is used for each beacon node to decide whether or not it is going to flood the network or use the limited to TTL flood is similar to the distributed way that nodes are selected as cluster heads in LEACH routing protocol [10]. The algorithm selects a fraction of the beacon nodes to flood the whole network, while the rest use random TTLs to disseminate the beacon's coordinates. Figure 10.2 demonstrates the process of the beacons flooding the WSN with a complete and limited (to the TTL) floods with the beacons' coordinates. The limited floods initiated by most of the beacons (represented by a black triangle) propagate the nodes inside the corresponding curved line, whereas the complete floods by the other beacons (represented by a white triangle) propagate the entire network.

An important feature of the proposed algorithm is that its phases may be run in parallel in the network, to utilize message exchange necessary in a following phase of the algorithm for piggybacking information from a previous phase and thus reducing communication cost. Thus, when a message containing an average hop distance traverses the network, it is piggybacked with the information about known beacon nodes and possibly the initial position estimate of a sender node. This way, the number of messages exchanged between the nodes is reduced.

The second way in which the LIP proposes to randomize the process of localization is that in the refinement phase of the algorithm, the nodes select random waiting periods for correcting their position estimates based on the information received from their neighbouring nodes. Weighted Moving Average is used when

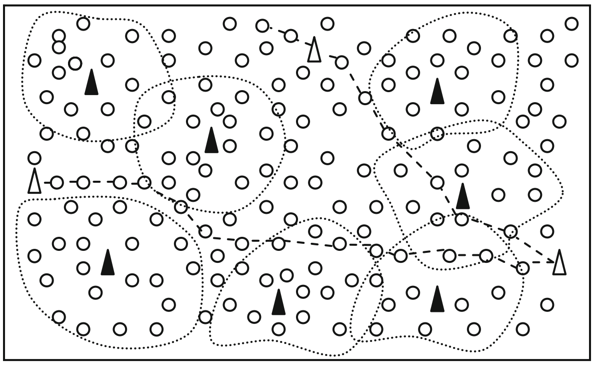


Fig. 10.2 Beacons use complete and limited floods to propagate their coordinates in the LIP [28]

the nodes receive multiple position corrections from a neighbouring node during the waiting period to emphasize the corrections with a high confidence. In addition, in the refinement phase, the algorithm employs low-duty cycling for the nodes that have low confidence in their position estimates. The goal of this scheme is to first rely on the nodes that are more confident in their position estimates to settle on their position estimates. Also, the rationale behind this is to ensure that the nodes that are not confident in their position estimates do not affect the lateration results of neighbouring nodes with their potentially erroneous position estimates. Meanwhile, the nodes using low-duty cycling conserve energy and avoid computation.

10.2.5.6 Comparison of Features of the Localization Algorithms

There exists a trade-off between the overhead of localization algorithms and precision in node localization that they provide. The algorithms using only the connectivity information (hop-based algorithms) have a lower overhead but the precision is compromised. On the other hand, the algorithms using ranging techniques may require additional hardware, which increases the overhead (the cost and size). However, this also increases the precision of node localization.

The *DV-hop* algorithm [30] uses only the connectivity information to localize nodes. *DV-distance* [30] and AHLoS [40], on the other hand, use ranging techniques such as RSSI and ToA to obtain distances between the nodes. The RPA [37] and LIP [28], on the other hand, use both the connectivity information for the initial position estimation and the ranging techniques for refining the position estimates iteratively. LIP proposes a lightweight solution to DV-hop-based algorithms by eliminating the synchronicity of the algorithms by means of randomizing the phases of localization.

Multilateration is used in the approaches RPA, LIP, and AHLoS to improve upon position estimates, while the method *n-Hop Multilateration* [39] considers multilateration that spans over multiple hops. The discussed features are summarized in Table 10.1.

Table 10.1 Comparison of features of the discussed localization algorithms

Algorithm	Connectivity information	Ranging techniques	Iterative
DV-hop	*		
RPA	*	*	*
AHLoS		*	
n-Hop multilateration	*	*	*
LIP	*	*	*

10.3 Temporal Event Ordering in WSNs

Temporal event ordering has generally been in focus of research on distributed systems. Event ordering for WSNs is a topic of active research as well. The problem of event ordering is concerned with the ordering of events (messages) in the chronological order of their occurrence. The necessary condition for resolving the problem of correctly ordering the events is to ensure that the nodes in the network are synchronized, i.e. every message is timestamped and the timestamps are accurate throughout the network. However, in order to ensure the correct ordering of the events that are received by a Sink, it is also necessary to make sure that there are no messages still in transit at the time when the messages are ordered.

The causal ordering in distributed systems is based on employing a “happened before” relation presented in [25]. Causal ordering algorithms (e.g., [33]) ensure that if $event_1$ happened before $event_2$, then $event_1$ will be processed before $event_2$ despite the possible violation of the order of the events’ reception at the receiver’s site. Causal ordering employs logical clocks; no physical clocks are used. WSNs often require the exact time at which the events occurred, and causal ordering is thus not sufficient for WSN applications.

10.3.1 Delaying Techniques

In *Delaying Techniques* [26, 43], there is an assumption about the upper bound of network delay D that messages can suffer. This time is equal to the maximum time it takes for a message to be sent by one of the nodes and received by another. There is a waiting time equal to that delay before the messages are ordered at a receiver’s site (such as a Sink). The receiver node keeps a list of the messages received. After a time equal to D, the first message in the list is removed and handed out to the application. These techniques may not suit WSNs because the delay in these networks is highly variable and it is thus difficult to choose an optimal value for D. If a longer time interval is set before the events are ordered, there will be some unnecessary idle time that will be wasted on waiting at the receiver. A shorter delay, on the other hand, could result in missing some messages that may still be in transit in the network (because it takes more time than D to arrive at the destination node).

10.3.2 Heartbeat

The *Heartbeat* [18] protocol employs the FIFO property of communication channels to ensure that all previously sent messages have been received before the events are ordered. This is achieved by having every node send a message in time intervals of *delta*. These messages are the control information to ensure that there are no delayed messages in the network. A receiver node (Sink) keeps an ordered list of the received messages. After a message with a timestamp greater than the timestamp of a received message is received at the receiver node from every node in the network, the first message in the list is removed and handed out to the application. A message m_1 that was received at the Sink node at time t_1 is removed from the list. It is handed out to the application, when the receiver node has received from every node in the network a message m_i with the timestamp $t_i > t_1$. Due to the FIFO property, all messages prior to the message m_1 have been received by the Sink before the messages m_i.

Like in *Delaying Techniques*, it is difficult to choose an optimal value for the time interval *delta*. If a small interval is chosen then the overhead will be too large, whereas if a large interval is used, then the system will be idle for a long time. Another drawback of the scheme is that whether or not there are messages generated in the network, the control messages from every node will be regularly sent to the Sink.

10.3.3 Temporal Message Ordering Scheme

Another event ordering algorithm for WSNs that is based on the FIFO property is the *Temporal Message Ordering Scheme* (TMOS) [35]. It constructs logical rings in the network. After the logical ring is constructed, when a node has a message to send, it sends the message in both directions of the logical ring. When both copies of the message are received by a node, it means that all previously sent messages have gone through that node due to the FIFO property of communication channels between the nodes in the network.

As shown in Fig. 10.3, a logical ring including the nodes $\{1, 2, 3, 4\}$, connected with solid lines, is constructed. When node 3 senses an event e_3 at time t_3, it sends two copies of the message: m_3 and $m_3^{'}$ to its neighbouring nodes, each of the messages is sent in one direction. After receiving the message, the receiver node 1 will first send all the events that were sensed locally and then send the message m_3 to its neighbouring node 4. Similarly, node 2 will send message $m_3^{'}$ after first sending locally sensed events to the neighbouring node 4 in the logical ring. Node 4 will insert the received events in a list ordered on the timestamps. Due to the FIFO property of communication channels, node 4 will receive at least one copy of all the sensed events which happened before the timestamp t_3 of the event e_3. After receiving the second copy of the message m_3, node 4 removes the first message of the ordered list and hands it out to the application.

The concept of this method is interesting, but it basically doubles the network traffic: for every message, there are two copies that are sent along the logical ring.

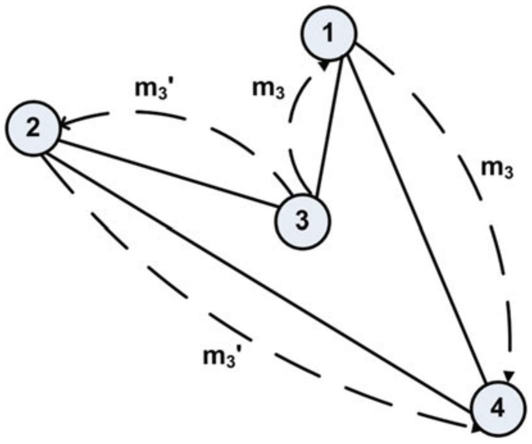


Fig. 10.3 TMOS algorithm (adopted from [35])

For energy-constrained WSNs this will imply a lot of energy dissipation. Also, construction of logical rings is not straightforward.

10.3.4 Ordering by Confirmation

The *Ordering by Confirmation* (OBC) [7] event ordering method for Wireless Sensor Actor Networks (WSANs) also uses the FIFO property of channels. This method first constructs a hop tree around the Actor, as is required for the routing protocol used for message delivery [8]. When the Actor (depicted as a black triangle in Fig. 10.4) needs to order events, it sends a temporal acknowledgment (ACK) request to the leaf nodes (the nodes at the path extremity) in the network. The leaf nodes are shaded in Fig. 10.4. These nodes then send temporal ACK messages towards the Actor. When these messages are received from all the nodes closest to the Actor, this ensures that all earlier messages have already been received. Next, the first message in the ordered list that is kept at the Actor node is removed and handed out to the application. This method does not require the nodes in the network to periodically send messages to the Actor. Only after the Actor broadcasts a temporal ACK request do the nodes in the network send temporal ACKs. Thus, the OBC does not suffer from the non-determinism of delay.

In the OBC, a node on a branching path does not forward the temporal ACK reply message before it has received the temporal ACK replies from all its children nodes. This is done in order to reduce the number of messages sent towards the Actor node. In addition, the Actor node maintains buffers for unacknowledged messages. It is possible that during the time when the Actor node waits to receive temporal ACKs regarding message m_1 (as shown in Fig. 10.4) that is generated by node A, it receives message m_2 generated by node B. If nodes on the path were not yet traversed by the temporal ACKs, then, both messages m_1 and m_2, ordered according to their corresponding timestamps, will be removed from the list and handed out to

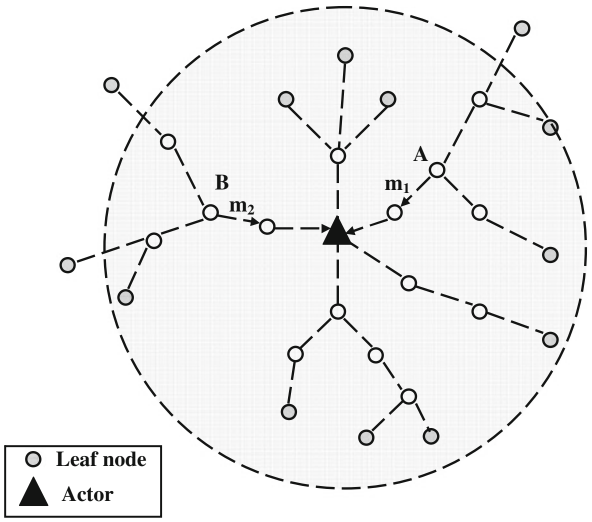


Fig. 10.4 OBC algorithm's temporal acknowledgment process (adopted from [7])

the application. This way, more than one message can be temporally acknowledged with a single temporal ACK request. This results in a reduced number of messages and, consequently, conserves the energy of WSNs. Otherwise, if the nodes were already traversed by temporal ACKs for the message m_1, the message m_2 is placed into a different buffer. It will be handed out to the application after the Actor sends another temporal ACK request and receives acknowledgments from the nodes in the network.

10.3.5 An Efficient Algorithm for Preserving Events' Temporal Relationships in Wireless Sensor Actor Networks

The approach presented in [5] proposes an algorithm to deal with the problems of temporal event ordering and time synchronization as a whole as both problems are concerned with preserving the temporal relationships of the events sensed in WSNs. Modifications to the OBC [7] algorithm are proposed. At the same time, a hybrid synchronization scheme for the clustered topology is proposed (the synchronization part of the algorithm is discussed in Sect. 10.4).

The proposed modifications to OBC consider introducing clustering concept into network topology (we will refer to the approach as *Clustered Ordering by confirmation* (COBC) hereafter). As discussed in Sect. 10.3.4, when there is a branching path in the paths created for message delivery in OBC, the parent node waits to receive the temporal ACK messages from its children nodes before forwarding it further in order to reduce the number of messages of the temporal acknowledgment process as compared with Heartbeat protocol. In the clustered topology, clustering is more general than a branching path because generally most of the nodes belong

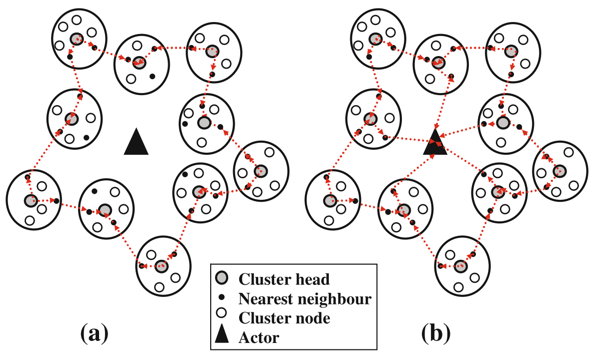


Fig. 10.5 (**a**) Initiation of the temporal ACK. (**b**) ACK replies reach the Actor

to clusters. Thus, a cluster head node will wait to receive a temporal ACK message from its cluster nodes and then forward one acknowledgment message on behalf of all its cluster nodes. Clustering can further reduce the overhead of event ordering in terms of delay and energy dissipation. In a clustered topology, only a cluster head (CH) node waits to receive the flush messages from its cluster nodes before sending the message further toward the Actor. If there are m nodes in a cluster, the CH will wait to receive temporal ACK messages from all the nodes and will then forward one message on behalf of all these nodes. This decreases the delay of temporal event ordering process. It also results in a reduced energy dissipation. In periodic event ordering approaches, if these nodes did not belong to a cluster, however, all m nodes would have to send a separate message toward the Actor.

Figure 10.5a shows the clusters located at the extremities of the area of the Actor's monitoring that initiate sending the temporal ACKs after hearing the temporal ACK request message from the Actor (depicted as a black triangle). In Fig. 10.5b, the temporal ACKs reach the Actor node by traversing the network using the underlying routing algorithm [6]. The latter is based on inter-cluster communication that is achieved by means of using nearest neighbour nodes between neighbouring clusters. At this time the Actor will order the messages according to their timestamps.

10.3.6 Comparison of Features of the Temporal Event Ordering Algorithms

Table 10.2 presents a comparison of features of the discussed algorithms on temporal event ordering in WSNs. While *Delaying Techniques* [26, 43] and *Heartbeat* [18] algorithms suffer from the *non-determinism of delay*, the other algorithms discussed in this chapter overcome this problem. The delay in WSNs is varying, and thus it is difficult to predict an upper bound of delay D in the former algorithm and the time interval *delta* in the latter algorithm. The *FIFO property* of communication channels is employed in numerous algorithms to tackle the problem of temporal event ordering in WSNs [5, 7, 35].

Table 10.2 Comparison of features of the discussed algorithms on temporal event ordering

Algorithm	Non-determinism of delay	FIFO	Periodic
Delaying techniques	*		
Heartbeat	*	*	*
TMOS		*	
OBC		*	
COBC		*	

In the *Heartbeat* algorithm, the control messages are sent periodically to ensure that the straggler messages are delivered to the Sink before the Sink orders the events. This creates an overhead as the control messages may traverse the network even when no events were sensed in the network. In the remaining algorithms on event ordering discussed in this chapter, however, Sink/Actor initiates the temporal acknowledgment request only after it has received an event notification from sensor nodes (as in OBC and COBC). In TMOS, the process necessary for event ordering is also initiated only when a sensor node has an event notification message to forward towards the Sink: two copies of the message are sent along the logical ring. Thus, the overhead associated with the process of event ordering is reduced.

10.4 Time Synchronization in WSNs

Time synchronization is essential for any distributed system and it is necessary in WSNs for performing data fusion, low-duty cycling, temporal event ordering and other purposes. Synchronization algorithms for traditional networks, such as Network Time Protocol (NTP) [29] used for keeping nodes in the Internet synchronized, are not well suited for WSNs. NTP uses a hierarchical scheme for time synchronization, in which a server synchronizes a number of clients. The servers in turn are synchronized by means of external sources (such as GPS). Thus, all the nodes in the Internet are synchronized to a global timescale. In [11], the authors discuss the differences between traditional networks and WSNs. The main concern that makes NTP unsuitable for time synchronization in WSNs is energy efficiency. As already noted above, in Sect. 10.2, GPS is unsuitable for combining with sensor nodes due to its energy consumption and cost. Also, it requires a line of sight to GPS satellites that is not always possible in WSN applications. Among other reasons that suggest refraining from using a global timescale-based synchronization are the absence of infrastructure in WSNs and the presence of high dynamics.

Figure 10.6 demonstrates the components of packet delay over a wireless link, which were first introduced in references [23] and [22] and then extended in reference [13]. We briefly describe each of the components below:

- *Send Time* — the time that is spent by the sender to construct a message, i.e. the highly variable delays introduced by the operating system caused by the syn-

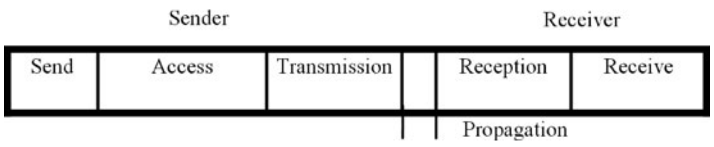


Fig. 10.6 Components of packet delay over a wireless link (adopted from [13])

chronization application (e.g., for context switches). The *Send Time* also includes the time necessary for the message to be transferred from the host to its network interface.

- *Access Time* — the time the packet waits for accessing the channel. The delay is also highly variable as it depends on the MAC protocol that is used (e.g., in TDMA channels the sender will have to wait for its time slot to transmit).
- *Transmission Time* — the time it takes the packet to be transmitted bit by bit at the physical layer over a wireless link. This delay component is deterministic as it depends on the radio speed.
- *Propagation Time* — the time that it takes the packet to propagate from the sender to the receiver. This delay component is very small and is negligible compared to the other components.
- *Reception Time* — the time spent on receiving the packet bit by bit at the receiver node. This component of the delay is deterministic in nature like the *Transmission Time* component.
- *Receive Time* — the time it takes to construct the packet and pass it to the application layer. This delay component is highly variable like the *Send Time* as it is affected by highly variable delays introduced by the operating system.

The resulting delay of a packet over a wireless link is thus highly affected by the delay associated with the MAC layer on the sender's side as well as the variable delays introduced by the operating system on both the sender's and receiver's sides.

10.4.1 Time Synchronization Techniques

The synchronization techniques that are used in synchronization algorithms for WSNs discussed later in this section can be divided into two broad categories. In the first category falls the traditional sender–receiver-based technique, while in the second one falls the receiver–receiver-based synchronization technique. Each one of the techniques is described in the following subsections.

10.4.1.1 Round-Trip Synchronization

In *Round-Trip Synchronization* (RTS) [34], the receiver of a sync pulse is synchronized with the sender. Two versions of the RTS technique could be used. In the first one, the receiver of a sync pulse replies to the sender right away. This way, the sender node can compute the round-trip delay and thus synchronize the receiver

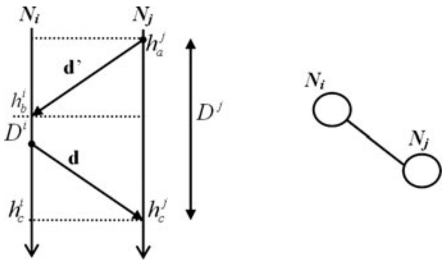


Fig. 10.7 Round-trip synchronization (adopted from [45])

node's time with its own. In the second version, the receiver of the sync pulse does not need to reply to the sync pulse right away, but only later on. It computes the delay between the reception of the sync pulse and the time when the reply is sent and attaches the delay information to the reply. The receiver node, upon receiving the reply, will subtract the delay time from the time of the reply's reception. The receiver node will thus be synchronized with the sender.

Figure 10.7 demonstrates the message exchange required for RTS. In this version of RTS, node N_j sends a request to node N_i about the timestamp at which the message was received $\left(h_b^i\right)$. Node N_i does not, however, reply to N_j right away but sends the reply message after some time D^i. After the reply is received by node N_j, it calculates the round-trip time (taking into account the delay D^i in sending the reply message).

10.4.1.2 Reference Broadcast Synchronization

Reference Broadcast Synchronization (RBS) [14], unlike traditional synchronization techniques in which a server synchronizes a set of clients with itself, synchronizes a set of receivers with each other. A beacon node broadcasts a message to the set of receivers using the network's physical layer broadcast. The receivers use the time of the message's arrival for comparing their clocks. Due to the nature of broadcast [17], a message that is sent at the physical layer will be received by the receiver nodes at approximately the same time. There are a few versions of RBS. In the main version, a beacon node broadcasts a message to two nodes in a network. Then the receiver nodes exchange messages with their local time (timestamp) at the time of the sync pulse's reception. This way, each node will have information about the other node's local time. The two receiving nodes will be synchronized with each other but not with the sender. In another version of RBS, the nodes receiving the reference broadcast send their timestamps of the sync pulse's reception back to the beacon node. This version of RBS is used when two nodes cannot communicate with each other [12, 14].

Figure 10.8 demonstrates the version of RBS in which, after two nodes receive a sync pulse from a beacon node, they send messages back to the beacon node with their timestamps at the moment of the sync pulse's reception. The nodes N_i and N_j reply to the beacon node N_k with messages containing their local times at the

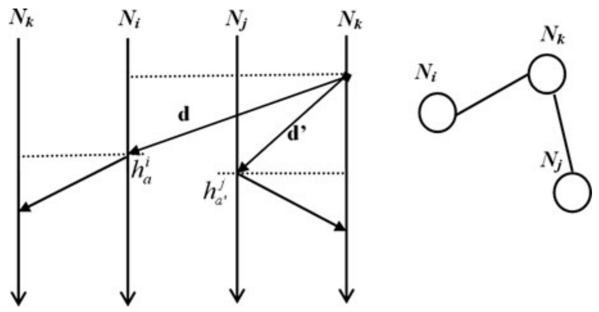


Fig. 10.8 The RBS method's version, in which local times at the moment when the sync pulse arrives at nodes N_i and N_j, is kept at the beacon node N_k

moment when the pulse was received (h_a^i for node N_i and $h_{a'}^j$ for node N_j). Thus, the beacon node will store the information about the differences in local times.

10.4.2 Synchronization Algorithms for WSNs

10.4.2.1 Multi-hop Time Synchronization

The *Multi-hop Time Synchronization* [14] uses RBS synchronization technique and relies on the use of "gateway" nodes—the nodes that are aware of the local timescales in more than one broadcast domain. In Fig. 10.9, each one of the nodes inside a single broadcast domain, such as nodes 1–3, 5, 6, 10, 11, can hear broadcasts with synchronization pulses coming from only one node labeled with a letter (such as A, B, C and D). On the other hand, nodes 4, 7, 8 and 9 can hear broadcast messages coming from two nodes. For instance, the nodes 8 and 9 can hear synchronization pulses coming from nodes C and D. These nodes are then used in the algorithm to perform multi-hop synchronization. The nodes are aware of the two local time scales and are able to convert the time scales and thus synchronize the other nodes within the corresponding broadcast domains. When a node senses an event E_i it timestamps it with its local time R_j. The notation $E_i(R_j)$ represents the time of event *i* according to the receiver *j*'s clock. Thus, to compare the time of event $E_1(R_1)$ with $E_{10}(R_{10})$ the algorithm will go through the following time conversions: $E_1(R_1) \to E_1(R_4) \to E_1(R_8) \to E_1(R_{10})$.

10.4.2.2 Timing Sync Protocol for Sensor Networks

The *Timing Sync Protocol for Sensor Networks* (TPSN) [13] is a hierarchical synchronization technique. There are two phases in the protocol: *Level Discovery* and *Time Synchronization*. In the first phase, a spanning tree is built. The root node of the spanning tree then broadcasts a *level_discovery* packet. The nodes are assigned a level in the hierarchy by propagating the broadcast of the root node. After the hop tree is built the second phase of the protocol starts. Pair-wise synchronization is

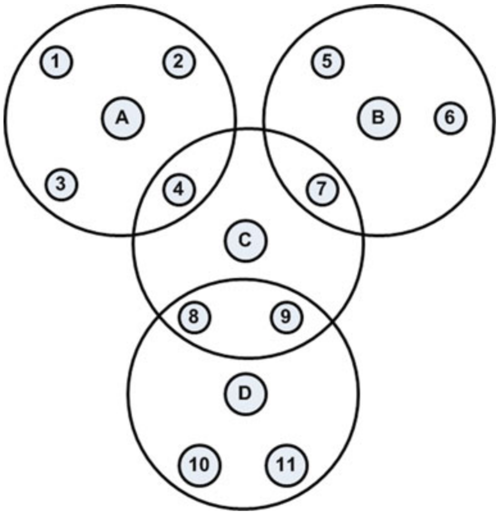


Fig. 10.9 The RBS method's multi-hop time synchronization

performed by the nodes along the edges of the hierarchical structure, i.e. every child node is synchronized with its corresponding parent node. The RTS synchronization technique is used by the nodes to achieve synchronization. Initially, the root node broadcasts a *time_sync* packet. After receiving this packet, the nodes in the first level wait for a random time and then send an acknowledgment to the root node. The procedure continues to the higher levels of the hierarchy of nodes. This way, every node in the network is synchronized to the root node. The important feature of the TPSN is that messages are timestamped at the MAC layer in order to reduce the delay variability by minimizing the *Send Time* component of delay. When network topology changes (for instance, due to node failures), the whole procedure of master election and tree construction will be repeated.

10.4.2.3 Flooding Time Synchronization Protocol

The *Flooding Time Synchronization Protocol* (FTSP) [27] is used for synchronizing a network to the root node as well. The node with the lowest ID is selected as a leader to be the reference point for time synchronization. The node periodically floods the network with messages containing its time. All the nodes that have not received the message record the timestamp contained in the message as well as the message's *time of arrival*. They then broadcast the message to the neighbouring nodes after updating the timestamp. Similar to TPSN [13], FTSP timestamps synchronization messages at the MAC layer as well. After a node has gathered eight pairs of the *timestamp* and *time of arrival*, the node uses linear regression on these data to obtain phase offset and clock skew with regards to the leader node.

10.4.2.4 Synchronization for a Clustered Topology

In the synchronization method proposed in reference [5] (we will refer to the method as *Synchronization for a Clustered Topology* (SCT) in this chapter), a scheme suitable for a clustered topology of Wireless Sensor Actor Networks (WSANs) (depicted in Fig. 10.5) is presented. It consists of the combination of the two synchronization techniques for WSNs — RBS and RTS. The cluster head (CH) nodes are synchronized with each other by means of the RBS, while the nodes inside clusters are synchronized with their corresponding CHs with RTS. The version of RBS depicted in Fig. 10.8 is used for keeping the CH nodes synchronized. The algorithm uses the Actor node as a beacon node that broadcasts the sync pulses to the CHs. The CHs in turn will reply to the Actor node with messages about their timestamps at the moment when the sync pulse was received.

The advantage of using RBS to synchronize CHs with each other, in this approach, is that the tables of phase offsets of the local times of the CHs are stored at the Actor. The Actor that has higher capabilities in terms of energy and computation performs all the time conversions necessary in order to synchronize the CH nodes with each other. This frees the rest of the energy-constrained nodes of the additional computations and storage. An example of a table with time offsets of the CH nodes that is kept at the Actor is presented in Table 10.3. The numbers in the table indicate the phase offset between two CH nodes. For instance, "-2" in the cell CH_1CH_2 indicates that the local time at the CH_1 is behind the local time at the CH_2 by 2 units of time (e.g., minutes).

Having the information above, let us assume that the Actor receives two events (E_1 and E_2) with their corresponding timestamps: E_1, timestamped at the CH_1 at the local time 12:02, and E_2, timestamped at the CH_2 at the local time 12:01. From these two events, we could conclude that E_2 happened before event E_1. However, after converting the local times at the Actor node, it is revealed that E_1 happened before E_2, as shown in Table 10.4.

Table 10.3 Table with phase offsets

	CH_1	CH_2	...	CH_n
CH_1	–	-2		1
CH_2	2	–		3
⋮			–	
CH_n	-1	-3		–

Table 10.4 Table with local times and converted times

CH	Local time	Time after conversion
CH_1	12:02	12:00
CH_2	12:01	12:01
CH_n	12:03	12:02 (when comp. with CH_1)
		12:00 (when comp. with CH_2)

Table 10.5 Table with local times and phase offsets

Node	Local time	Phase offset
Actor	12:00	
CH_1	12:02	+2
CH_2	11:59	-1
CH_n	12:03	+3

The method described above is sufficient when only relative synchronization is required (for instance, for timing the propagation delay of sound [15]), i.e. it is important to know which event occurred before or after a given event. For applications requiring absolute time, the method uses the version of RBS that employs synchronization with an external reference. If the Actor is equipped with a GPS, the CH's times are then converted with regard to the Actor's time. An example is shown in Table 10.5.

If the Actor receives three events, E_1 timestamped at the CH_1 (at the local time 13:01), E_2, timestamped at the CH_2 (with the local time 13:01) and E_3 timestamped at the CH_3 (with the local time 13:00), then after conversion the times will be as follows: E_1 12:59; E_2 13:00 and E_3 12:57. This way, the receivers of the RBS sync pulses are synchronized to an external time scale.

In the approach, cluster nodes and free nodes are synchronized by means of RBS synchronization technique. The synchronization pulses and replies are piggybacked on the message exchange necessary in the temporal event ordering and underlying routing algorithms. Thus, message exchange that is necessary for synchronizing the network is significantly reduced.

10.4.3 Comparison of Features of the Time Synchronization Algorithms

As seen above, the delay of a packet over a wireless link is highly affected by the *Send Time*, *Access Time*, and *Receive Time* (shown in Fig. 10.6). The traditional synchronization algorithms based on *sender–receiver* mode of time synchronization, such as RTS [34], in which a receiver node is synchronized with a sender node in WSNs, suffer from the uncertainty of these components of delay. The *receiver–receiver*-based synchronization technique such as RBS [14], on the other hand, minimizes the uncertainty of the delay by eliminating the *Send Time* and *Access Time* components from the total packet delay. This happens due to the fact that these components of delay are the same for all the receivers of the packet. The algorithms employing the RBS method, such as *Multi-Hop Time Synchronization* [14] and SCT [5], are thus able to decrease the synchronization error. The reference [14] also discusses the possibility of further reducing the total delay of a packet if it is timestamped at the low level in the receiver node host's operating system kernel. Therefore, the *Receive Time* will not include the overhead of system calls, context switches, etc.

Table 10.6 Comparison of features of the discussed algorithms on time synchronization

Algorithm	Sender–receiver	Receiver–receiver	Tunable
Multi-Hop RBS		*	
TPSN	*		
FTSP	*		
SCT	*	*	*

Another approach to the task of reducing the components of delay are followed by the synchronization protocols TPSN [13, 27] that proposed to timestamp messages at the MAC layer at both the sender's and receiver's sides. This way, delay variability is decreased, and consequently, the message delay uncertainty is decreased.

The SCT approach presents a hybrid solution to synchronization in WSANs, where both *receiver–receiver* (RBS) and *sender–receiver* (RTS) modes of synchronization are employed as discussed above. The algorithm follows the directions, identified in reference [11], about designing tunable synchronization algorithms. The synchronization techniques used in the method are chosen according to the communication type of the underlying event ordering and routing algorithms (for instance, as part of the event ordering algorithm, the Actor node broadcasts a temporal acknowledgment request, which in this method is piggybacked with the synchronization pulse to synchronize the cluster head nodes with each other by means of the RBS). This approach results in an overall lower overhead of time synchronization. A comparison of features of the algorithms is presented in Table 10.6.

10.5 Summary

In this chapter, we presented an overview of the algorithms concerning the spatio-temporal context in WSNs in three areas: node localization, temporal event ordering and time synchronization.

The localization algorithms for WSNs discussed in this chapter are more precise when not only connectivity information for localization purposes but also the ranging techniques are used. However, the overhead is thus increased as well. For DV-hop-based algorithms, reducing the communication cost is critical in order to resolve the scalability issue. The iterative algorithms attempt to increase the precision of node localization, by employing a refinement stage, after the initial position estimates are obtained by the nodes.

While some of the temporal event ordering algorithms for WSNs overviewed in this chapter suffered from the non-determinism of delay, the other algorithms have overcome this problem. In the more recent algorithms, in order to reduce communication cost, the necessary control messages (that ensure there are no straggler messages in the network before the Sink orders events) are sent by the nodes after a request from the Sink is received and not periodically, as it was done in the earlier algorithms.

The synchronization algorithms for WSNs overviewed in this chapter aimed at reducing various components of delay. While the sender–receiver-based synchronization technique suffers from the uncertainty of delay, the receiver–receiver-based synchronization technique achieves a smaller synchronization error by eliminating some of the components of delay noted above. MAC timestamping was used by some of the discussed algorithms in order to reduce the delay introduced by the MAC layer at the sender's and receiver's sides. Another direction of synchronization algorithms are tunable algorithms, which aim at selecting the synchronization techniques suitable for the communication mode of the underlying algorithms (such as a routing algorithm, for instance) and using piggybacking for synchronization pulses and replies. This results in a reduced communication overhead of time synchronization.

To conclude, with an ultimate goal of making WSNs "intelligent", it is necessary to take into account the context of the sensed events. A consideration of the spatio-temporal context in WSNs enables the building of a foundation for context-aware systems. Then, the concept of context can be extended past the limits of space and time.

References

1. J. Albowitz, A. Chen, and L. Zhang. Recursive position estimation in sensor networks. In *Proceedings of IEEE* ICNP, pages 35–41, 2001.
2. P. Bahl, and V. N. Padmanabhan. Radar: An in-building RF-based user location and tracking system. In *Proceedings of IEEE Infocom*, 775–784, 2000.
3. A. Boukerche. Algorithms and protocols for wireless sensor networks. *Wiley Series on Parallel and Distributed Computing*, 2008.
4. A. Boukerche, H. Oliveira, E. Nakamura, A. Loureiro. Vehicular ad hoc networks: a new challenge for localization-based systems. *Computer Communications* 31(12):2838–2849, 2008.
5. A. Boukerche, and A. Martirosyan. An efficient algorithm for preserving events' temporal relationships in wireless sensor actor networks. In *Proceedings of the 32nd IEEE Conference on Local Computer Networks*, pages 771–780, 2007.
6. A. Boukerche, and A. Martirosyan. An energy-aware and fault tolerant inter-cluster communication based protocol for wireless sensor networks. In *Proceedings of the 50th IEEE Global Telecommunications Conference*, pages 1164–1168, 2007.
7. A. Boukerche, F. H. S. Silva, R. B. Araujo, R. W. N. Pazzi. A low latency and energy aware event ordering algorithm for wireless actor and sensor networks. In *Proceedings of the 8th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 111–117, 2005.
8. A. Boukerche, R. W. N. Pazzi, and R. B. Araujo. A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications. In *Proceedings of 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 157–164, 2004.
9. H. Chan, M. Luk, and A. Perrig. Using clustering information for sensor network localization. *Lecture Notes in Computer Science*, Springer, Berlin, pages 109–125, 2005.
10. A. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.
11. J. Elson, K. Romer. Wireless sensor networks: a new regime for time synchronization. *ACM SIGCOMM, Computer Communication Review*, 33(1):149–154, 2003.

12. J. Elson. *Time Synchronization for Wireless Sensor Networks*, PhD Thesis, University of California, Los Angeles, 2003.
13. S. Ganeriwal, R. Kumar, M. B. Srivastava. Timing-sync Protocol for Sensor Networks. In *Proceedings of the 1st International Conference on Embedded networked sensor systems*, pages 138–149, 2003.
14. J. L. Girod and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the Fifth Symposium on Operating systems Design and Implementation*, pages 147–163, 2002.
15. L. Girod and D. Estrin. Robust range estimation using acoustic and multimodal sensing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1312–1320, 2001.
16. G. H. Golub and C. F. Van Loan. *Matrix Computations*, 3rd edition, Johns Hopkins University Press, 1996.
17. J. Y. Halpern and I. Suzuki. Clock synchronization and the Power of Broadcasting. *Distributed Computing*, 5(2):73–82, 1991.
18. R. Hayton. *OASIS: An Open Architecture for Secure Interworking Services*, PhD Thesis, University of Cambridge, Cambridge, 1996.
19. J. Hightower and G. Borriello. Location systems for ubiquitous computing, *Computer*, 34(8):57–66, 2001.
20. W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocols for wireless sensor networks. In *Proceedings of the 33rd Annual International Conference on System Sciences*, pages 3005–3014, 2000.
21. R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, Wiley, New York, 1991.
22. H. Kopetz, and W. Schwabl. *Global Time in Distributed Real-Time Systems*. Technische Universitat Wien, 1989.
23. H. Kopetz, and W. Ochsenreiter. Clock synchronization in distributed real-time systems. *IEEE Transactions on Computers*, C-36(8):933–939, 1987.
24. D. Koutsonikolas, S. M. Das, Y. C. Hu. Path planning of mobile landmarks for localization in wireless sensor networks. *Computer Communications*, 30(13):2577–2592, 2007.
25. L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 54(3):558–565, 1978.
26. M. Mansouri-Samani, M. Sloman. GEM a generalized event monitoring language for distributed systems. *IEE/IOP/BCS Distributed Systems Engineering Journal*, 4(25):96–108, 1997.
27. M. Maroti, B. Kusy, G. Simon, and A. Ledeszi. The flooding time synchronization protocol. In *ACM Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 39–49, 2004.
28. A. Martirosyan, and A. Boukerche. A lightweight iterative positioning algorithm for context-aware wireless sensor networks. In *Proceedings of IEEE Globecom*, 2009.
29. D. L. Mills. Internet time synchronization: the network time protocol. In Yang Z. and Anthony T. Marsland, editors, *Global States and Time in Distributed Systems*. IEEE Computer Society Press, 1994.
30. D. Niculescu and B. Nath. Ad hoc positioning system. In *Proceedings of the IEEE International Conference on Global communications (GlobeCom ' 01)*, pages 2926–2931, 2001.
31. B. Parkinson and J. Spilker. *Global Positioning System: Theory and Application*. American Institute for Aeronautics and Astronautics, 1996.
32. N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Mobile-assisted localization in wireless sensor networks. In *IEEE INFOCOM*, pages 172–183, 2005.
33. M. Raynal, A. Schiper, S. Toueg. The causal ordering abstraction and a simple way to implement it. *Information Processing Letters archive*, 39(6):343–350, 1991.
34. K. Römer, P. Blum, L. Meier. Time synchronization and calibration in wireless sensor networks, In Stojmenovic I., editor, *Handbook of Sensor Networks: Algorithms and Architectures*, Wiley, New York, NY, pages 199–237, 2005.

35. K. Römer. Temporal message ordering in wireless sensor networks. *IFIP Mediterranean Workshop on Ad-Hoc Networks*, pages 131–142, 2003.
36. K. Römer. Time synchronization in ad hoc networks. In *ACM Symposium on Mobile Adhoc Networking and Computing*, pages 0–1, 2001.
37. C. Savarese, J. M. Rabaey, K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks, In *Proceedings of the General Track: 2002 USENIX Annual*, pages 317–327, 2002.
38. C. Savarese, J. M. Rabaey, and J. Beutel. Locationing in distributed ad-hoc wireless sensor networks. In *ICASSP*, pages 2037–2040, 2001.
39. A. Savvides, H. Park, and M. Srivastava. The n-Hop multilateration primitive for node localization problems, *Mobile Networks and Applications*, 8:443–451, 2003.
40. A. Savvides, C. Han, M. B. Strivastava. Dynamic fine-grained localization in Ad-Hoc networks of sensors, In *Proceedings of the 7th annual International Conference on Mobile computing and networking*, pages 166–179, 2001.
41. B., Schilit, N., Adams, R.: Want, Context-Aware Computing Applications. In *Proceedings of the First International Workshop on Mobile Computing Systems and Applications*, pages 85–90, 1994.
42. Y. Shang, W. Ruml, Y. Zhang, M. Fromherz. Localization from connectivity in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 15(11):961–974, 2004.
43. Y. C. Shim and C. V. Ramamoorthy. Monitoring and control of distributed systems. In *Proceedings of the 1st International Conference of Systems Integration*, pages 672–681, 1990.
44. M. L. Sichitiu and V. Ramadurai. Localization of wireless sensor networks with a mobile beacon. *Technical Report. TR-03/06* Center for Advances Computing Communications, North Carolina State University, 2003.
45. I. Stojmenovic. *Handbook of Sensor Networks, Algorithms and Architectures*, Wiley, New York, NY, 2005.
46. R. Stoleru, J. A. Stankovic, S. H. Son. Robust node localization for wireless sensor networks. In *Proceedings of the 4th workshop on Embedded networked sensors*, 2007.
47. K.-F. Ssu, C.-H. Ou, and H. C. Jiau. Localization with mobile anchor points in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 54(3):1187–1197, 2005.
48. K. Whitehouse, D. Culler. Calibration as parameter estimation in sensor networks. In *Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications*, pages 59–67, 2002.
49. B. Xiao, H. Chen, and S. Zhou. A walking beacon-assisted localization in wireless sensor networks. *Proceedings of the IEEE International Conference on Communications (ICC '07)*, pages 3070–3075, 2007.

Chapter 11
Coordination Problems in Ad Hoc Radio Networks

Dariusz R. Kowalski

Abstract In this chapter we consider coordination problems in the model of wireless communication called *ad hoc radio network*. This model evolved from a multiple-access channel, which was introduced as a communication model for single-hop LANs, such as Ethernet. In radio networks, communication is assumed to be in (synchronous) slots, and an interference of two or more transmission signals received by a node results in a failure of delivering any of the colliding messages to this node. In ad hoc setting nodes are not aware, or have very limited knowledge, of the topology of the system, including the underlying network or distribution of active stations. Moreover, their local clocks may often show different readings. We consider several coordination problems in the context of ad hoc radio networks with no a priori given clock synchronization, such as waking up of sleeping nodes, unifying local clock settings, electing a leader, and mutual exclusion. We present the state of the art in these areas and suggest a few perspective research directions.

11.1 Introduction

It is a well-known fact that efficient wireless communication is endangered by various physical constraints (such as signal attenuation, reflection, and interference), by the impact of mobility of devices, by malicious attacks, and by many other factors. One of the ways of overcoming the impact of such obstacles, often unpredictable, is to impose additional coordination mechanisms for communication and computing in wireless environment. This chapter presents several examples of coordination and control problems, together with existing solutions, described and analyzed in the model of ad hoc radio networks.

D.R. Kowalski (✉)
Department of Computer Science, University of Liverpool, Liverpool L69 3BX, UK
e-mail: D.Kowalski@liverpool.ac.uk

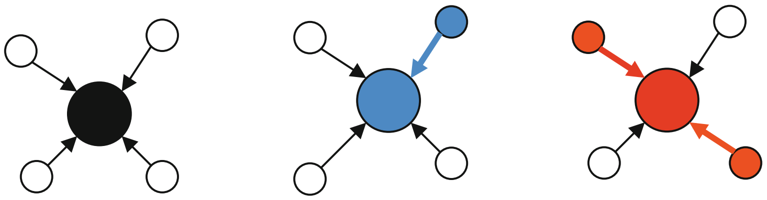


Fig. 11.1 Examples of three different feedbacks provided to a node in a transmission round. *Left-side* scenario: no in-neighbor transmits and the middle node hears nothing (background noise); *Middle-side* scenario: exactly one in-neighbor transmits and the middle node successfully receives the message transmitted by the in-neighbor; *Right-side* scenario: two in-neighbors transmit simultaneously and the middle node hears collision (signal interference)

11.1.1 Model and Problems

In radio networks, processing units, called stations or nodes, communicate by transmitting and receiving messages to/from their neighborhoods. Communication has a property that multiple messages arriving at the same time to a node interfere with one another and none of them can be successfully received by this node. Radio network is modeled as a directed strongly connected graph $G = (V, E)$, with $n = |V|$ nodes. We assume $V = [1 \dots n] = [n]$. Each station is equipped with a capability to transmit a message and to receive a message or other feedback resulting from simultaneous transmissions of other nodes. An edge (v, w) in graph G models the property that node w is in transmission range of node v, i.e., a message sent by node v can reach node w. A message sent by node v can be *heard by*, or *successfully delivered to*, node w if no other in-neighbor of node w, except v, is transmitting at the same round; if more than one neighbor transmits then an interference, also called a *collision*, is heard (cf. Fig. 11.1). If a node is equipped with a device distinguishing between background noise, also called a *silence* (i.e., when no neighbor transmits), and collision, then we call it a node *with collision detection*; otherwise it is a node *without collision detection*. In this work we focus on networks where all nodes do not have collision detection capability (though, as we mention later, some work has also been done in the alternative model where *all* nodes are with collision detection).

Size n is the only network parameter which can be a part of the input for protocols we consider. Apart from it, each node knows only its own distinct integer identified from $[n]$. We denote by D the maximum directed distance between a pair of nodes (i.e., the diameter), and by Δ we denote the maximum in-degree of a node.

11.1.1.1 Coordination Problems

In this work we survey solutions for several coordination problems. We start with the *wake-up* problem, in which the goal is to activate all nodes in the network. In the beginning, at least one node is active, while activation of the other nodes—initially sleeping—may happen either arbitrarily or by receiving a message from the other active node. Sleeping nodes do not perform any action (they neither do any local computation nor transmit). Arbitrary activation is scheduled by some function σ, called a *shift function* or an *activation function*, from the set of nodes to non-negative integers such that $\sigma_v = 0$ for at least one node v. This function is unknown to the nodes, in the sense that it is not provided to the node as a part of the input.

The other two coordination problems considered in this work are *leader election* and *clock synchronization*. In the leader election problem, we want to designate one node as a *leader* and to announce its identity to all nodes in the network. In the clock synchronization problem, upon the completion of the protocol, all nodes must have the same local clock setting.

All these problems—fundamental from the perspective of network communication and distributed computing—have been studied in the last decades in various different environments. They are all examples of one-instance coordination problems, i.e., they have to reach specified level of coordination only once. We also define and study a dynamic mutual exclusion problem—an example of "long-live" coordination problems in which requests for exclusive access to the channel arrive in a distributed and online fashion and must be accommodated by a distributed protocol run by the stations (see Sect. 11.5).

11.1.1.2 Complexity Measures

In this chapter we study time complexity of coordination problems, defined as the number of rounds counted from the first round when some station starts executing the protocol until the task is finished. Formal definition of the completion time depends on the particular coordination task. For the purpose of dynamic coordination problems, such as dynamic mutual exclusion, we will later introduce more suitable measure called makespan (see Sect. 11.5).

11.1.1.3 Assumptions and Notation

To avoid rounding, we assume that n is a power of 2. The notation $\log x$ denotes the logarithm of x to the base 2.

11.1.2 Results

We start (Sect. 11.2) with presenting the best up-to-date algorithms for deterministic and randomized wake-up problem on single-hop networks; they are based on the results developed in [9, 14, 15, 20, 38]. In Sect. 11.3 we provide extensions of these protocols to multi-hop networks, based on the result from [14, 20]. The solutions of clock synchronization and leader election can be found in Sect. 11.4; they are based on the results from [14, 20]. Mutual exclusion problem is formally defined and analyzed in Sect. 11.5, based on the results from [9, 25]. Section 11.6 contains final remarks and states several perspective research directions. Below we give a summary of results concerning the considered problems and other related coordination tasks.

11.1.2.1 Wake-Up Problem and Radio Synchronizers

Wake-up protocols were first considered for the multiple-access channel by Gasieniec et al. [30]. Such networks are also called single-hop radio networks and are modeled by complete directed graphs (note that for this network topology wake-up, leader election and synchronization are equivalent). One of the results is a deterministic protocol accomplishing wake-up in $O(n \log^2 n)$ rounds. The proof of this fact was existential. Randomized wake-up protocols for the multiple-access channel were studied by Jurdzinski and Stachowiak [38]. A generalization of the wake-up problem for multi-hop networks was first studied by Chrobak et al. [20]. They introduced the notion of radio synchronizer—a binary array of n rows with the property that for any admissible row shift[1] there is a column with exactly one entry 1—and also developed leader election and synchronization protocols. In particular, they showed that there are (n, k)-synchronizers of length $O(k^2 \log n)$ and used them to solve wake-up problem in multi-hop networks in $O(n^{5/3} \log n)$ rounds. Generic transformations of wake-up protocols into leader election and synchronization algorithms shown in [20] add a logarithmic factor to the time complexity. Chlebus and Kowalski [15] introduced the notion of universal radio synchronizers to improve synchronization algorithms. They showed that for each n there is an (n, f)-universal synchronizer with delay $f(n, k)$ upper bounded by the function $O\left(k \min\left\{k, \sqrt{n}\right\} \log n\right)$. The fastest known deterministic wake-up protocol, given by Chlebus et al. [14], is based on improved upper bound $O(k \log k \log n)$ on delay function $f(n, k)$ in universal synchronizers. The resulting existential wake-up solution for multi-hop networks works in time $O(\min\{n, D\varDelta\} \log \varDelta \log n)$.

Clementi et al. [22] showed that slightly simpler structures, called (n, k)-*selective-families*, have to be of a size $\varOmega(k \log(n/k))$. (More precisely, selective families have the properties of radio synchronizers for the special case where all rows are shifted by 0.) It follows that (n, k)-synchronizers have to be of length $\varOmega(k \log(n/k))$.

A construction of (n, n)-synchronizers of length $O(n^{1+\varepsilon})$, for any constant $\varepsilon > 0$, in a quasi-polynomial time $O(2^{\text{polylog}\, n})$ was given by Indyk [36]. Chlebus and Kowalski [15] specified explicit (n, k)-synchronizers of length $O(k^2 \operatorname{polylog} n)$. Prior to this work, there was no known non-trivial explicit construction of *universal* synchronizers. Chlebus et al. [14] constructed explicit (n, f)-universal synchronizers with $f(n, k)$ of order $O(k^2 \operatorname{polylog} n)$. This was the first explicit construction of universal synchronizers. The resulting explicit solution for multi-hop networks works in time $O(\min\{n, D\varDelta\}\varDelta \operatorname{polylog} n)$. This is the first *explicit sub-quadratic* wake-up solution for $\varDelta = o(n/ \operatorname{polylog} n)$.

Two-way synchronizers were introduced by Chlebus et al. [17] under the name of radio transmitters. The authors showed that there exists n-two-way synchronizers of length $O(n \log^2 n)$ and gave a polynomial-time construction of n-two-way syn-

[1] Here by "admissible row shift" we mean a specified class of row shifts; in this sense, the actual definition of a synchronizer depends on a class of row shifts that we allow in the definition; more details and a few examples of different classes of synchronizers can be found in Sect. 11.2.1.

chronizers of length $O(n^2 \log n)$. Two-way synchronizers were applied as a tool for design-efficient dynamic acknowledgment-based broadcast protocols on a multiple-access channel (see also Sect. 11.1.2.4).

A slightly different problem of wake-up of anonymous undirected radio network was considered by Pelc [48]. In this problem, there is a distinguished node initializing the wake-up process, while all other nodes are indistinguishable and without any additional knowledge. It was shown that this problem is feasible if the communication is scheduled in synchronous slots, while it is not feasible if transmissions can be delayed up to some factor $t > 1$ in the worst-case manner.

11.1.2.2 Leader Election, Clock Synchronization, and Other Related Problems

Most of the solutions for these problems developed in the context or radio networks assumed that all nodes are awake from the beginning. For deterministic leader election on the multiple-access channel without collision detection, matching bounds on time $O(n \log n)$ and $\Omega(n \log n)$ follow from [22, 44], with the upper bound being non-constructive. A constructive upper bound $O(n \text{ polylog } (n))$ follows from [36]. For the expected time of randomized algorithms without collision detection, the same matching bounds are known: $\Omega(\log n)$ follows from [45] and $O(\log n)$ from [6]. Leader election problem was extensively studied in the model *with collision detection*. For the time of deterministic algorithms with collision detection, matching bounds are also known: $\Omega(\log n)$ follows from [31], and $O(\log n)$ follows from [11, 34, 53]. Randomized leader election can also be done faster than in the model without collision detection: matching bounds $\Omega(\log \log n)$ (for fair protocols) and $O(\log \log n)$ were proved in [54]. Further references can be found in [37, 47]. Extensions to multi-hop radio networks have also been considered, cf. [21].

The existing leader election and synchronization protocols in our model have time performance slower than wake-up by an additional factor of $O(\log n)$, due to the generic transformation given in [20].

Farach-Colton and Mosteiro [28] studied so-called group therapy problem in Weak Sensor Model, which is a model with geometric topologies and uncoordinated wake-ups. Group therapy problem is weaker than the synchronization problem, it only assumes that each node receives successfully a message from one of its neighbors. The lower bound $\Omega(\Delta + \log \Delta \log(1/\varepsilon))$ for the time complexity established in [28], where ε is the probability of error, holds as well for synchronization problem.

11.1.2.3 Wake-Up in the Channel with Many Frequencies

Alonso et al. [3] studied static wake-up problem in the context of node discovery in ad hoc networks. They considered randomized protocols with fixed probabilities of transmission in each frequency. They developed formulas for expected time complexity of node discovery, expressed in terms of the fixed probabilistic distribution.

Dolev et al. [26] considered wake-up problem in the channel with many frequencies; some of them could be temporarily disrupted. Their model of radio network consists of F disjoint narrowband communication frequencies, where $n \geq F$. Each round, every node selects a single frequency on which to participate, where participation means either transmitting or receiving. An interference adversary was considered that could disrupt up to $t < F$ frequencies per round, where t is a known upper bound. By disrupting a frequency, the adversary prevents any node from receiving a message on that frequency. A node receives a message on a frequency f only if exactly one node broadcasts on f, and the adversary does not disrupt f. The adversary chooses its behavior for round r based only on knowledge of the protocol being executed and the completed execution up to the end of round $r - 1$. The authors proved that every regular protocol, i.e., a protocol in which devices behave in a uniform fashion, achieving wake-up with probability at least $1 - 1/n$ requires $\Omega\left(\frac{\log^2 n}{(F-t)\log\log n} + \frac{Ft}{F-t}\log n\right)$ rounds. They also gave the Trapdoor Protocol that almost matches the lower bound, i.e., uses $O\left(\frac{\log^2 n}{(F-t)\log\log n} + \frac{Ft}{F-t}\log n\right)$ rounds with high probability. The modified version of this protocol was shown to adapt to the number of available frequencies if all stations start in the same time. More precisely, in such scenarios it needs only $O(t' \log^3 n)$ rounds to synchronize if at most $t' \leq t$ frequencies can be disrupted at a time and all participating stations start in the same round (t' is not a part of the input), and $O(F \log^3 n)$ rounds otherwise; all these results hold with high probability.

11.1.2.4 Dynamic Control Problems and Dynamic Broadcast

Kowalski [41] and Bender et al. [8] were the first who considered communication in the multiple-access channel when packets are injected into the system by an adversary. They followed different approaches. In [41], deterministic protocols based on acknowledgment are studied under the assumption that there are no more than k active stations in a round. Bender et al. [8] studied randomized backoff-type protocols under queue-free adversarial queuing setting. These two early works have initiated two slightly different research directions: dynamic mutual exclusion (control-type problem) and dynamic broadcasting (communication-type problem).

Czyzowicz et al. [25] studied deterministic solutions for mutual exclusion problem in the multiple-access channel. They developed an algorithm achieving makespan $O(n \log^2 n)$, which, in view of the lower bound $\Omega(n)$, is close to optimal. They also studied the impact of the knowledge of parameter n, global clock, and collision detection on feasibility and efficiency of mutual exclusion. In particular, the authors proved that with none of those three characteristics mutual exclusion is infeasible. On the side of efficiency, they presented an optimal—in terms of the makespan measure—$O(\log n)$ algorithm in the model with collision detection.

In [9] randomized solutions for the mutual exclusion problem and its "almost-always" version, called ε-exclusion, on the multiple-access channel were considered. The main focus was in fairness, unlike the previous paper [25] where only no-deadlock property was guaranteed. The authors proved an exponential gap between solutions for the classic mutual exclusion and ε-exclusion. More precisely, it was shown that the expected makespan of algorithms solving the mutual exclusion problem is $\Omega(n)$, while there is an algorithm with expected makespan $O(\log n \cdot \log 1/\varepsilon)$. They also presented a generic transformation of a mutual exclusion algorithm with no-deadlock property into the one satisfying stronger no-lockout condition. This method applied to the deterministic algorithms from [25] produces efficient deterministic solutions satisfying no-lockout property. Similar to the previous paper [25], other model settings were also considered in this context.

The problem of broadcasting dynamically injected packets on the multiple-access channel also involves several coordination aspects. The main properties to be guaranteed are stability (bounded number of packets in the system) and fairness (each packet is eventually considered). Different variants of these objectives have been considered, depending on the model of packet injection. Early work in this direction included developing randomized protocols such as Aloha [1] and (binary) exponential backoff [46]. The analysis of broadcasting process focused on scenarios when packets are injected subject to *statistical constraints*; see [29] for an overview of early research and [35, 51] for more recent results. Bender et al. [8] studied stability of randomized backoff on the multiple-access channel in the *adversarial queuing model*.[2] In this setting, packet injection is modeled by adversaries and does not involve any stochastic component in the process of packet generation. An adversary is limited by the rate of injecting packets and by the burstiness of traffic. Chlebus et al. [17] studied deterministic protocols for broadcasting dynamically injected packets on the multiple-access channel. They considered the *queuing model*, in which packets are injected into stations communicating via the channel, unlike in [8], where packets were injected directly to the system as "independent stations" (*queue-free model*).

11.1.2.5 Related Research Directions

General radio networks were intensely studied since the seminal paper [15] analyzing broadcasting in this model. Many different communication tasks have been considered, such as broadcasting (cf. [2, 6, 13, 22, 24, 43, 45, 50]), gossiping (all-to-all, cf. [24]), many-to-many (cf. [7, 16]), in various different settings, though most of them assumed fully synchronized environment (so-called static case). Some of the algorithmic techniques developed in these works have been later upgraded to the model without clock synchronization and for the online setting.

Communication with possible failures (such as crashes, Byzantine faults, probabilistic failures) has also been studied in the context of radio networks, see, e.g., [23, 27, 40, 49]. In [19] feasibility and complexity of agreement in a multiple-access channel with *synchronized* starting points and crash failures were studied in the context of different collision detectors—the tools introduced in that work by

[2] Adversarial Queuing Theory (AQT) was introduced by Borodin et al. [10] in the context of store-and-forward networks.

the analogy to classic failure detectors. Asynchronous *centralized* communication in radio networks was studied by Chlebus and Rokicki [18].

11.2 Wake-Up on a Multiple-Access Channel

Coordination problems are not easily solved even in a relatively simple model of single-hop networks. This section presents the best known deterministic and randomized solutions to the wake-up problem. Since it is not known how to construct efficiently such deterministic solutions, we conclude by giving an example of polynomial-time construction of a deterministic wake-up algorithm with slightly worse properties.

11.2.1 Deterministic Synchronization

A deterministic protocol waking up a single-hop network can be viewed as a binary array satisfying specific properties. In the protocol associated with such an array, node v simply follows the pattern defined by row v of the array: it transmits in every round corresponding to the position with entry 1 in this row and only listens in the remaining rounds. A single row in this array is also called a *transmission sequence*. The number of columns in this structure, also called its length, is the upper bound on the time complexity of the corresponding wake-up protocol. Such arrays are called *radio synchronizers*, or *synchronizers* for short. We present three types of synchronizers: (n, k)-synchronizers, (n, f)-universal synchronizers, and n-two-way synchronizers, where $k \leq n$ is a positive integer and f is a function from $[n]$ to positive integers. We also argue about existence of efficient synchronizers of these three types, by using a randomized construction and applying the probabilistic argument. The reason for considering several different properties of wake-up arrays is their possible applicability for multi-hop networks and for other coordination-type problems.

We will look at a collection of transmission sequences as a binary array of n rows and m columns, where rows correspond to stations and columns correspond to communication rounds. A *shift function* σ is a function from the set of nodes to non-negative integers,[3] corresponding to different wake-up times of nodes in the channel. We say that a position, or a communication round, t is successful, if there is exactly one transmitting active node on the channel in round t (i.e., there is exactly one node v among those shifted by less than t which has entry 1 on the position $t - \sigma_v$).

[3] In case of strong synchronization, e.g., for two-way synchronizers, σ may be subject to slightly different constraints.

11.2.1.1 Synchronizers and Universal Synchronizers

The basic synchronizers are defined as follows:

> Synchronizers:
> A binary $n \times m$ array $\mathcal{S}$ is a (n, k)-*synchronizer*, for given integer parameters $n, m > 0$ and integer $1 \leq k \leq n$, if for any shift function σ and any set K of k rows, with the property that $\sigma_K = \min_{v \in K} \sigma_v = 0$, there is a successful position for K between positions 1 and m.

An example of a $(4, 4)$-synchronizer and its shift by function $\sigma_1 = 0, \sigma_2 = \sigma_3 = 2, \sigma_4 = 3$ is given in Fig. 11.2.

Synchronizers are structures defined to handle scenarios when the number of competing stations is arbitrarily bounded by parameter k. To allow more flexibility, the notion of synchronizers was generalized as follows.

> Universal synchronizers:
>
> A binary $n \times m$ array $\mathcal{S}$ is a (n, f)-*universal-radio-synchronizer*, for given integer parameters $n, m > 0$ and integer function $f \in ([n] \times [n])^{[m]}$, if
>
> - $f(n, n) = m$, and
> - for any shift function σ and any set L of rows, with the property that $\sigma_L = \min_{v \in L} \sigma_v = 0$, there is a successful position for L between positions 1 and $f(n, |L|)$.

We call function f a *delay function*. Our goal is to show existence of a universal synchronizer with relatively small delay function. The proof is by using the probabilistic method.

Random universal synchronizers We start with specifying length m and delay function f for which we define a random universal synchronizer. Let $c \geq 4$ be a constant to be determined later. Define $m(\ell) = c2^{\ell}\ell \log n$, for every $1 \leq \ell \leq \log n$. Let $m = \sum_{\ell \leq \log n} m(\ell)$. We define $f(n, k)$ by $\sum_{\ell \leq \lceil \log k \rceil} m(\ell)$ for $k > 1$ and $f(1)$ by $m(1)$. Observe that $f(n, k) = O(k \log k \log n)$. It also follows that for each j, where $1 \leq j \leq m$, there is a unique positive integer $\kappa_j \leq \log n$ satisfying the inequalities

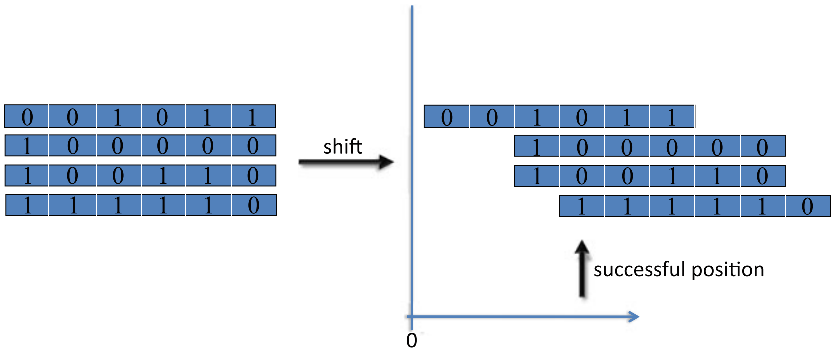


Fig. 11.2 Example of $(4, 4)$-synchronizer of length 6. After applying an arbitrary non-negative shift σ to the set of rows (in this case $\sigma_1 = 0, \sigma_2 = \sigma_3 = 2, \sigma_4 = 3$), there must be a column (number four in this example) with exactly one entry

$\sum_{\ell<\kappa_j} m(\ell) < j \le \sum_{\ell\le\kappa_j} m(\ell)$ (here, in order to avoid cases in the definition, we assume that $\sum_{\ell<1} m(\ell) = 0$).

Consider a random $n \times m$ array $\mathcal{S} = [\mathcal{S}(v, j)]$, where the binary-valued random variables $\mathcal{S}(v, j)$ are all independent, for $1 \le v \le n$ and $1 \le j \le m$, and each random variable $\mathcal{S}(v, j)$ is defined to be equal to 1 with probability $1/(\kappa_j 2^{\kappa_j})$.

Theorem 1 (Chlebus et al. [14]) *The array $\mathcal{S}$ is an (n, f)-universal synchronizer, for $f(n, k) = O(k \log k \log n)$, with the probability of at least $1 - 1/n$, for a sufficiently large constant $c \ge 4$.*

Proof Fix a positive integer n. We will use shorter notation $f(k)$ for $f(n, k)$, since n is fixed throughout the proof. For a position t, let $A_\sigma(t)$ be the set of rows σ-active at t; we write $A(t)$ when σ is clear from context. We partition shift functions into n classes in such a way that σ *belongs to class* $\mathcal{C}_i$ if i is the smallest number such that the inequality $|A_\sigma(f(i))| \le i$ holds. These classes are well defined because all rows are active at position m by the assumption that $\sigma_v < m = f(n)$, for $1 \le v \le n$.

Our goal is to show that for a shift function σ in class $\mathcal{C}_i$, there is a successful position by the position $f(i)$ in synchronizer $\mathcal{S}$ with rows shifted by σ. To prove this, it is sufficient to consider only a partial function, denoted by $\sigma|^{f(i)}$, defined by the values of σ at rows v such that $\sigma_v < f(i)$.

We first consider shift functions in class $\mathcal{C}_1$. By definition of the class, there is only one row v active on positions $\{1, \ldots, 2c \log n\}$. The probability that all these positions in row v are unsuccessful is at most $2^{-2c \log n} \le e^{-c \log n}$. Since there are at most n possible rows v to be considered, the probability that there is a shift function in class $\mathcal{C}_1$ with all positions $t \le f(1) = 2c \log n$ unsuccessful is at most $e^{\ln n} \cdot e^{-c \log n} \le e^{-2 \ln n} = n^{-2}$, since $c \ge 4$.

Next, consider the general case of shift functions in class $\mathcal{C}_k$, for $1 < k \le n$. Let us fix a shift function σ in this class. By definition, number k is the smallest positive integer such that $|A_\sigma(f(k))| \le k$ holds. To simplify notation in the remainder of the proof, we denote set $A_\sigma(f(k))$ by K. For row $v \in K$ and position $t \le f(k)$, let $p(v, t)$ denote the probability that there is 1 in shifted row v at position t, i.e., the probability that $\mathcal{S}(v, t - \sigma_v) = 1$, with $p(v, t)$ set to 0 for every $t \le \sigma_v$. Let $\mu_K(t)$ denote $\sum_{v\in K} p(v, t)$.

We call a position $t \le f(k)$ *balanced* if the inequalities $1/(4\kappa_t) \le \mu_K(t) \le 4$ hold. Let κ stand for $\kappa_{f(k)}$. Observe that $\kappa \ge \kappa_t$ and $m(\kappa) = f(k) - f(k/2)$. We show that a balanced position is a successful position, with probability $\Omega(1/\kappa)$.

Lemma 1 (Chlebus et al. [14]) *A balanced position $t \le f(k)$ is successful with probability at least $\frac{1}{4^5\kappa}$.*

Proof The probability that position $t \le f(k)$ is successful can be estimated from above as follows:

$$\sum_{v\in K} p(v, t) \cdot \prod_{w\in K, w\ne v} (1 - p(w, t)) \ge \mu_K(t) \cdot 4^{-\mu_K(t)} \ge \frac{1}{4\kappa} 4^{-4} = \frac{1}{4^5\kappa}$$

since $p(v, t) \le 1/2$ for every $v \in K$. □

In the following lemma we estimate the number of balanced positions in the interval $[f(k/2)+1, f(k)]$.

Lemma 2 (Chlebus et al. [14]) *There are at least* $m(\kappa)/2$ *balanced positions in the interval* $[f(k/2)+1, f(k)]$.

Proof We have $|A(f(k/2))| > k/2$ and $A(f(k)) = K$ has k elements, since k is the smallest integer i such that $|A(f(i))| \leq i$. By changing the order of summation we obtain

$$\sum_{t \leq f(k)} \mu_K(t) = \sum_{v \in K} \sum_{t \leq f(k)} p_v(t)$$

For every $v \in A(f(k))$ we have

$$\sum_{t \leq f(k)} p_v(t) \leq \sum_{\ell \leq \kappa} m(\ell) \cdot \frac{1}{\ell 2^{\ell}} = c\kappa \log n$$

Consequently, an estimate

$$\sum_{v \in K} \sum_{t \leq f(k)} p_v(t) \leq c|K|\kappa \log n$$

holds, which implies

$$\sum_{t \leq f(k)} \mu_K(t) \leq ck\kappa \log n$$

By the pigeonhole principle, there are at least $3f(k)/4$ positions $t \leq f(k)$ such that

$$\mu_K(t) \leq 4 \cdot \frac{ck\kappa \log n}{f(k)} \leq 4$$

where the last inequality follows from the fact that $f(k) \geq m(\kappa) \geq ck\kappa \log n$. Consider the positions t such that $f(k/2)+1 \leq t \leq f(k)$. Since

$$k/2 < |A(t)| \leq k$$

and

$$p_v(t) \geq \frac{1}{\kappa 2^{\kappa}}$$

we have that

$$\mu_K(t) \geq |A(t)| \cdot \frac{1}{\kappa 2^{\kappa}} > (k/2) \cdot \frac{1}{2k\kappa} = \frac{1}{4\kappa}$$

Now we put all these partial results together. Since

$$f(k) - f(k/2) = m(\kappa) \geq f(k)/2$$

we obtain that at least

$$3f(k)/4 - f(k/2) = f(k) - f(k/2) - f(k)/4 = m(\kappa) - f(k)/4 \geq m(\kappa)/2$$

positions t in the interval $[f(k/2) + 1, f(k)]$ satisfy inequality $\mu_K(t) \leq 4$, and consequently at least $m(\kappa)/2$ positions in the interval $[f(k/2) + 1, f(k)]$ are balanced. This completes the proof of Lemma 2. □

We resume the main proof of Theorem 1. By Lemmas 1 and 2, the probability that all balanced positions not bigger than $f(k)$ are unsuccessful is at most

$$\left(1 - \frac{1}{4^5\kappa}\right)^{m(\kappa)/2} \leq e^{-(2c/4^6)2^\kappa \log n} \leq e^{-(2c/4^6)k \log n}$$

The above considerations were for a fixed shift σ from class $\mathcal{C}_k$. We have to show the existence of a successful position $t \leq f(k)$ for *any* shift σ in class $\mathcal{C}_k$ and later for *any* class. In order to do it, we have to estimate the number of shift functions that are pairwise "different" from the point of view of their behavior (and thus their analysis). Note that we may skip from our consideration all rows shifted by σ by more than $f(k)$ and focus only on different configurations defined by shifts from class $\mathcal{C}_k$ in positions $\{1, \ldots, f(k)\}$. In particular, if two shifts in class $\mathcal{C}_k$ have the same shift values for rows shifted by at most $f(k)$, they observe the same behavior by position $f(k)$ and may be represented by the same configuration in positions $\{1, \ldots, f(k)\}$. Each different configuration defined by a shift from $\mathcal{C}_k$ in positions $\{1, \ldots, f(k)\}$ is determined by its domain of k rows and an assignment of shift values in the range from 0 to $f(k) - 1$ to them. It follows that there are at most $\binom{n}{k} \cdot (f(k))^k \leq e^{k \ln(ne/k) + k \ln f(k)}$ different configurations defined by shifts from class $\mathcal{C}_k$ in positions $\{1, \ldots, f(k)\}$.

The probability that there exists a shift function σ from the class $\mathcal{C}_k$ such that there is no successful position $t \leq f(k)$ in $\mathcal{S}$ shifted by σ is at most

$$e^{k \ln(ne/k) + k \ln f(k)} \cdot e^{-(2c/4^6)k \log n} \leq e^{-2k \ln n}$$

for a sufficiently large constant $c \geq 4$, since $\ln f(k) = O(\log n)$.

Summing up all the probabilities for all classes, the probability of the event that there is a shift function in *some* class $\mathcal{C}_k$ such that there is no successful position $t \leq g(k)$ in $\mathcal{R}$ is at most $\sum_{k=1}^{n} e^{-2k \ln n} \leq 1/n$, for $n > 1$. This completes the proof of Theorem 1. □

Deterministic universal synchronizers By applying probabilistic argument to Theorem 1, we obtain the following existential result.

Corollary 1 *There exists a deterministic (n, f)-universal synchronizer for some delay function $f(n, k) = O(k \log k \log n)$.*

11.2.1.2 Toward Stronger Synchronization: Two-Way Synchronizers

In case of (universal-) synchronizers, we restricted to shifts with non-negative values and at least one row to be non-shifted. It modeled a scenario when there is a clear "starting point" of wake-up execution. It may not be the case in general executions, therefore in this section we study executions without clear starting point, i.e., a shift function σ has integer values, not necessarily positive, as it was required for (universal-) synchronizers. Our goal is to argue that there is a successful position in a period of sufficient "activity" of stations/rows. A formal definition follows.

Consider a binary array $\mathcal{S}$ of n rows and m columns and an arbitrary shift function σ of its rows (i.e., a function from the set of rows to integers). We say that a position $i \in \{1, \ldots, m\}$ is *covered* by a shifted row v in $\mathcal{S}$ if row v is shifted by at least $i - m$ and at most $i - 1$ positions, i.e., $i - m \le \sigma_v \le i - 1$. We say that the shift function σ is *proper* when both conditions hold:

(i) the inequality $-m + 1 \le \sigma_v \le m$ holds, for each row i, and
(ii) each position in $\{1, \ldots, m\}$ is covered by at least one row in array $\mathcal{S}$ shifted by σ.

Note that the above definition does not depend on the specification of the entries of $\mathcal{S}$, but only on its size. We call a position $i \in \{1, \ldots, m\}$ in array $\mathcal{S}$ shifted by σ *successful* if there is a row v in $\mathcal{S}$ such that shifted by σ_v row v has entry 1 at position i while the other rows of $\mathcal{S}$ shifted by σ covering position i have entries 0 at position i.

Two-way synchronizers:
A binary $n \times m$ array $\mathcal{S}$ is a *n-two-way synchronizer* of length m, for given integer parameters $n, m > 0$, if for any proper shift function σ there is a successful position in $\mathcal{S}$ shifted by σ between positions 1 and m.

An example of a 4-two-way synchronizer and its shift by the function $\sigma_1 = -4$, $\sigma_2 = \sigma_3 = -2$, $\sigma_4 = -1$ is given in Fig. 11.3.

Similar to the case of universal synchronizers, existence of "good" two-way synchronizers can be shown using the probabilistic method. We define a random $n \times m$ array $\mathcal{S}$ with $m = cn \log^2 n$, for a sufficiently large constant $c > 0$. Let $\mathcal{S}(v, i) = 1$ with probability $1/n$, independently over all choices of row v and column i. It can be shown (cf. [17]) that $\mathcal{S}$ is a n-two-way synchronizer of length m with a positive probability. Consequently, by the probabilistic argument, we obtain:

Theorem 2 (Chlebus et al. [17]) *There exists a n-two-way synchronizer of length $O(n \log^2 n)$, for any positive integer n.*

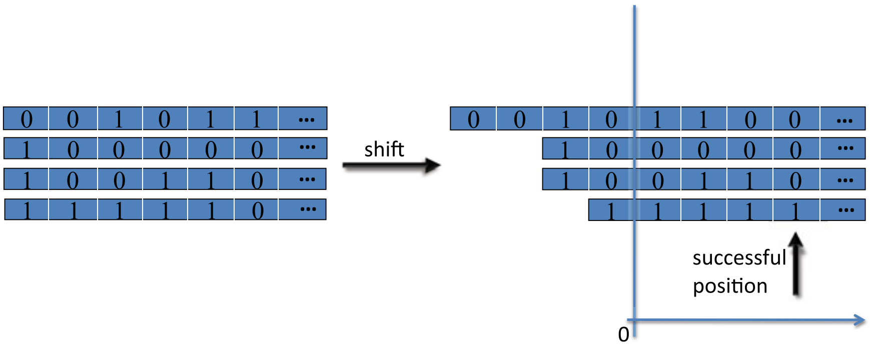


Fig. 11.3 Example of 4-two-way synchronizer of length m. After applying an arbitrary shift σ to the set of rows (in this case $\sigma_1 = -4, \sigma_2 = \sigma_3 = -2, \sigma_4 = -1$), there must be a column (number four in our case) with exactly one entry 1 among the columns with number between 1 and m

11.2.2 Randomized Synchronization

Randomized wake-up algorithms appear to be much more efficient than the deterministic ones. The following algorithm IncreaseFromSquare [38] is an example of efficient application of move-on paradigm in the context of wake-up problem.

Algorithm 1 IncreaseFromSquare

```
$k \leftarrow \lceil \log_{32/31}(1/\varepsilon) \rceil$
$y \leftarrow \pi^2/6$
$f_v \leftarrow \lceil \log(2y(v+1)^2) \rceil$
$p_v \leftarrow 1/2^{f_v}$
Upon activation, node $v$ performs the following while $p_v \leq \frac{1}{2}$:
  1. if $p_v \leq 1/2$ then in each of the next $k$ rounds node $v$ transmits with probability $p_v$, independently
  2. $p_v \leftarrow 2p_v$
```

We show that algorithm IncreaseFromSquare [38] solves even stronger T-*persistent selection problem* [9], for some function $T : \mathbb{N} \cup \{0\} \to \mathbb{N}$. This stronger property can be particularly useful in applications to other coordination type of problems (e.g., dynamic mutual exclusion [9]).

T-persistent selection problem:

We are given a function $T : \mathbb{N} \cup \{0\} \to \mathbb{N}$. A T-persistent selection problem is defined as follows. Arbitrary nodes are activated by an adversary in arbitrary rounds. Let t_v denote the round in which node v is started. Any solution for the T-persistent selection problem has to guarantee the following two properties:

1. Node v is allowed to transmit only in rounds $t_v, t_v + 1, \ldots, t_v + T(v) - 1$;
2. A successful transmission occurs in some round t satisfying the following condition: for any node v activated before round t, it holds that $t \leq t_v + T(v) - 1$.

In the analysis of algorithm IncreaseFromSquare the following technical fact is used. It can be proved (cf. [38]) using standard calculus.

Lemma 3 (Jurdzinski and Stachowiak [38]) *We are given k numbers $p_1, p_2, \dots, p_k \in \left(0, \frac{1}{2}\right]$ such that $\frac{1}{2} \le \sum_{i=1}^{k} p_i \le 3/2$. Then*

$$\sum_{i=1}^{k} p_i \prod_{\substack{j=1 \\ j \ne i}}^{k} (1 - p_j) \ge \frac{31}{32}$$

Theorem 3 (Bienkowski et al. [9] and Jurdzinski and Stachowiak [38]) *For any $0 < \varepsilon < 1$, algorithm* IncreaseFromSquare *solves the T-persistent selection problem for $T(v) = \Theta(\log v \cdot \log(1/\varepsilon))$, with probability at least $1 - \varepsilon$.*

Proof Let $T(v) = k \cdot f_v$. Observe that $T(v) = \Theta(\log v \cdot \log(1/\varepsilon))$. Note that node v executes the algorithm only for $T(v)$ consecutive rounds.

Let $\mu(t)$ denote the sum, over all nodes, of transmission probabilities p_v taken in round t. Consider the first round, say t_1, in which at least one of the nodes is active. We have

$$\mu(t_1) < \sum_{v=0}^{\infty} \frac{1}{2^{f_v}} = \sum_{v=0}^{\infty} \frac{1}{2^{\lceil \log(2y \cdot (v+1)^2) \rceil}} \le \sum_{v=1}^{\infty} \frac{1}{2y} \cdot \frac{1}{v^2} = \frac{1}{2}$$

It follows from the specification of the algorithm that $\mu(\cdot)$ is monotonically non-decreasing in time. Let t_2 be the last round t with the property $\mu(t) < 1/2$. Since transmission probabilities may only double every k rounds until reaching the threshold value $\frac{1}{2}$, we get that $1/2 \le \mu(t_2 + 1) < 1$. Observe that the transmission probabilities are always powers of $1/2$, thus, at round t_2, if node v is active then $p_v \le 1/4$. In particular, it follows that no node becomes inactive before round t_2.

Now we consider a time period R consisting of rounds $t_2 + 1, t_2 + 2, \dots, t_2 + k$. To finish the proof, we need to show that both conditions in the definition of T-persistent selection are satisfied. Indeed,

1. Nodes that were active at round t_2 are still active at least till round $t_2 + k$, since none of them reached the probability threshold $1/2$ by round t_2, and thus must continue for at least k more rounds.
2. In every round t of period R we have $1/2 \le \mu(t) \le 3/2$. This is because of the following four facts: First, $1/4 \le \mu(t_2) < 1/2$. Second, $\mu(t_2 + 1) \ge 1/2$. Third, each station active in round t_2 changes (i.e., doubles) its probability only once in a period of k contiguous rounds, in particular, in R, and thus these stations may only double their contribution to $\mu(t)$ compared to $\mu(t_2)$, for any $t \in R$. Fourth, the sum of transmission probabilities of nodes which are activated during period R is smaller than $\sum_{v=0}^{\infty} 1/2^{f_v} \le 1/2$. By Lemma 3, the probability that there is no successful transmission during the whole period R is at most $(31/32)^k \le \varepsilon$.

□

Note that algorithm IncreaseFromSquare does not need n as an input, and therefore Theorem 3 holds even if n is unknown to the nodes.

11.2.3 Explicit Constructions

In this section we show how to construct (n, k)-synchronizer with length $O(k^2 \text{ polylog } n)$ in time polynomial in n. This construction was given by Chlebus and Kowalski [15]. Based on their construction, Chlebus et al. [14] developed explicit (n, f)-universal-synchronizers with delay function $f(n, k) = O(k^2 \text{ polylog } n)$.

11.2.3.1 Construction of (n, k)-Synchronizers

Assume that number n is sufficiently large and that k divides n. We construct a (n, k)-synchronizer $\mathcal{S}$. Let P be a set of k different primes between $k \log n$ and $3k \log n$. Set P is well defined by the Chebyshev Theorem, as given in [5]. Consider a prime p in P. We denote by π_p a binary sequence of length $3k^2 \log n$ such that $\pi_p(j) = 1$, when p divides j, and $\pi_p(j) = 0$ otherwise. Let $\hat{\pi}$ denote a sequence of zeros of length $3k^2 \log n$. Let h be a function from $[n] \times [d]$ to P, where parameter d is a positive integer, to be determined later, depending on numbers k and n and on applications. For $v \in [n]$, define row $\mathcal{S}(v)$ of $\mathcal{S}$ to be a concatenation of schedules of the form $\langle \pi_{h(v,1)}, \hat{\pi}, \pi_{h(v,2)}, \hat{\pi}, \ldots, \pi_{h(v,d)}, \hat{\pi}, \sigma^{2d} \rangle$, where $\hat{\pi}^{2d}$ denotes a sequence of zeros of length $6dk^2 \log n$ (i.e., sequence $\hat{\pi}$ concatenated $6d$ times). In order to assure synchronization property of the constructed array $\mathcal{S}$, function h must have specific properties. We show that selective functions are good for this purpose.

Selective functions:
Function $h : [n] \times [d] \to P$ is said to be (n, k, d)-*selective* when for every set $W \subseteq [n]$ of a size at most k, there is a number p in P such that set $h^{-1}(p) \cap (W \times [d])$ is a singleton.

Lemma 4 (Chlebus et al. [15]) *If function h is (n, k, d)-selective, then the constructed array $\mathcal{S}$ is a (n, k)-synchronizer of length $m(n, k) = O(dk^2 \log n)$.*

Indyk [36] gave an explicit (n, k, d)-selective function for $d = \text{polylog } n$. His construction relies on explicit $(n, k', d', 1/2 - \varepsilon)$-dispersers developed by Ta-Shma et al. [52], for some parameter d' polylogarithmic in n and for a constant $0 < \varepsilon < 1/2$. Using this explicit selective function and Lemma 4, the main result can be shown:

Theorem 4 (Chlebus et al. [15]) *A family of (n, k)-synchronizers of length $m(n, k) = O(k^2 \text{polylog} n)$, where $1 \le k \le n$ are integers, can be constructed in time polynomial in n.*

11.3 Wake-Up in Multi-hop Radio Networks

Wake-up solutions described in Sect. 11.2 in the context of a multiple-access channel can be used as a tool in a design of efficient solutions for multi-hop radio networks. The key idea of such extensions is based on the concept of *path graphs*, i.e., subgraphs consisting of a path and some specific edges incoming to this path (cf. [42]). This idea occurs in design and analysis of many multi-hop protocols, which use single-hop protocols as a tool, cf. [20, 42]. Recall that D denotes the diameter of the directed multi-hop network, and Δ is its maximum in-degree.

11.3.1 Deterministic Wake-Up

The wake-up problem can be solved by a deterministic algorithm in time $O(n \operatorname{polylog} n)$, by using universal synchronizers shown to exist. When explicitly constructed universal synchronizers are applied, the wake-up time increases to $O(n\Delta \operatorname{polylog} n)$. Recall that universal synchronizers were developed and used in the context of single-hop networks, cf. Sect. 11.2.

11.3.1.1 Wake-Up Protocol Based on Universal Synchronizer

Let $\mathcal{S}$ be a (n, f)-universal synchronizer, where rows correspond to nodes. Each node v starts executing its sequence $\mathcal{S}(v)$, being the vth row in $\mathcal{S}$, immediately after being activated (spontaneously or by receiving a message). Recall that executing a transmission sequence means that node v performs a transmission in the ith round exactly when the ith bit in $\mathcal{S}(v)$ is a 1. We refer to this protocol as CK.

For numbers n, D, Δ and a delay function f of a (n, f)-universal synchronizer, let $\beta(n, D, \Delta, f)$ denote the supremum of the function $\sum_{i=1}^{D} f(x_i)$, where integers $0 \le x_i \le \Delta$, for $1 \le i \le D$, satisfy an additional constraint $\sum_{i=1}^{D} x_i \le n$. We show that β is an upper bound for the wake-up time achieved by protocol CK.

Theorem 5 (Chlebus et al. [15]) *Protocol* CK *based on* (n, f)*-universal synchronizer wakes up a radio network of n nodes, diameter D, and maximum in-degree Δ in $\beta(n, D, \Delta, f)$ rounds.*

Proof Let function σ determine the times of spontaneous wake-ups in the execution of algorithm CK. Note that activation function σ is closely related to shift functions considered in the context of single-hop networks, cf. Sect. 11.2. Let v_0 be among the nodes that become awaken spontaneously first.

In the analysis we use *path graphs* P_v, for every node $v \neq v_0$, defined as follows. The graph P_v contains some shortest path from v_0 to v, together with all in-neighbors of nodes on the path and the edges from them to nodes on the path.

For a node $v \neq v_0$, we distinguish a shortest path $\langle v_0, v_1, \ldots, v_L = v\rangle$ of length $L \le D$ in P_v and call it the *main path of* P_v. Let v_i be a node on this path. Let $\delta(v_i)$ denote the number of in-neighbors w of node v_i in graph P_v with

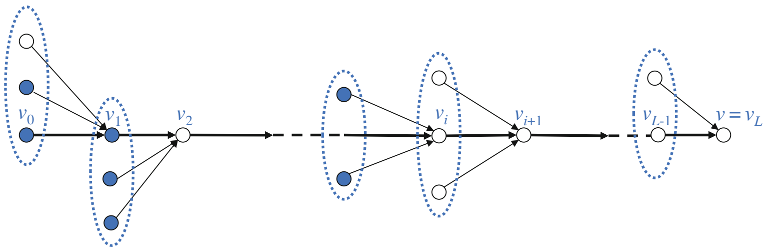


Fig. 11.4 Path graph P_v and activation status in some round t. $\langle v_0, v_1, \ldots, v_j, \ldots, v_L = v\rangle$ is the main path of P_v. *Blue* color denotes active nodes, while the *white* nodes are asleep; here v_i is the front node

the property that w neither follows v_i on the main path nor is it joined to any node following v_i on the main path, in the sense of the order of indices.

For a round t, there is a unique node v_i on the main path $\langle v_0, v_1, \ldots, v_j, \ldots, v_L\rangle = v$ with the following two properties:

(a) v_i is not active in round t, but one of its in-neighbors in P_v is active in round t,
(b) node v_j is not active in round t and none of its in-neighbors in P_v is active in round t, for every $j > i$.

We refer to the index i of this v_i as the *front value* $\psi(t)$ and to v_i as the *front node in round t* (cf. Fig. 11.4). Notice that a sequence of front values is non-decreasing in any execution. The following fact describes the progress made on the main path of P_v.

Fact 1 (Chlebus et al. [15]) *If* $\psi(t) < L$ *and* $\delta(v_{\psi(t)}) = k$, *then* $\psi(t + f(k)) > \psi(t)$.

Consider any node v and its graph P_v of $L+1$ nodes on the main path. It follows from Fact 1 and from the definition of (n, f)-universal synchronizer that node v becomes active by round

$$\sum_{1\le i\le L} f(\delta(v_i))$$

Each node w in P_v contributes only once to the sum $\sum_{i=1}^{L} \delta(v_i)$, namely to this term $\delta(v_i)$ where v_i has the property that w is an in-neighbor of v_i and w is not an in-neighbor of a node v_k for any $k > i$. Therefore, the inequality

$$\sum_{1\le i\le L} \delta(v_i) \le n$$

holds. Applying the upper bound Δ on $\delta(v_i)$ we get

$$\sum_{i=1}^{L} f(\delta(v_i)) \leq \beta(n, D, \Delta, f)$$

which completes the proof of Theorem 5. □

Applying the existential and explicit constructions of (n, f)-universal synchronizers given in [14] (see also Sects. 11.2.1 and 11.2.3) for functions $f(n, k)$ being $O(k \log k \log n)$ and $O(k^2 \operatorname{polylog} n)$, respectively, we get the following results.

Corollary 2 *Protocol* CK *can be instantiated such that it solves the wake-up problem in* $O(\min\{n, D\Delta\} \log \Delta \log n)$ *rounds.*

Corollary 3 *Protocol* CK *can be* explicitly *instantiated such that it solves the wake-up problem within time bound* $O(\min\{n, D\Delta\}\Delta \operatorname{polylog} n)$.

The estimate given in Corollary 2 is within a logarithmic factor away from $\Omega(\min\{n \log D, D\Delta \log(n/\Delta)\})$, which is a lower bound on broadcasting given in [22] that holds also for wake-up.

11.3.2 Randomized Wake-Up

We present a randomized wake-up solution for multi-hop networks that is based on algorithm IncreaseFromSquare [38] presented in Sect. 11.2.2. More precisely, each node starts running slightly modified version of IncreaseFromSquare algorithm just after being awakened. There are two changes in the algorithm compared to the single-hop version:

- instead of starting with its own id, it starts with n;
- it takes parameter ε/n as an error, so that the multi-hop solution could guarantee correctness with probability at least $1 - \varepsilon$.

We refer to this algorithm as MultihopIncreaseFromSquare, or MIFS for short. Note that one could replace the IncreaseFromSquare component in MIFS by another efficient wake-up solution for single-hop networks and hope to obtain an efficient modification of the original MIFS.
Combining

- the path-graph methodology used in the analysis of deterministic wake-up protocols in the proof of Theorem 5 and
- the fact that the progress along the main path between the first activated node and a given node v is in time $O(\log n \log(n/\varepsilon))$ with probability at least $1 - \varepsilon/n$,

we obtain the following result:

Theorem 6 (Chrobak et al. [20]) *Let* $\varepsilon > 0$, *let* G *be a multi-hop radio network with* n *nodes, and let* σ *be an arbitrary wake-up function. With probability at least* $1 - \varepsilon$, *algorithm* MIFS *completes wake-up in network* G *under wake-up function* σ *in time* $O(D \log n \log(n/\varepsilon))$.

Algorithm MIFS can be used to obtain a Las Vegas protocol with low expected running time. The protocol has three stages. First, algorithm MIFS is run for $\varepsilon = 1/n^3$. Second, each active node suspends itself for T rounds, where T is the upper bound on time complexity of MIFS from Theorem 6. Finally, a deterministic wake-up protocol, such as CK described in Sect. 11.3.1, is executed. We refer to the new protocol as MIFS-LV.

Theorem 7 (Chrobak et al. [20]) *Protocol* MIFS-LV *completes wake-up in radio network G of n nodes, diameter D, and maximum in-degree Δ, under a wake-up function σ in expected time $O(D \log^2 n)$ and worst-case time $O(\min\{n, D\Delta\} \log \Delta \log n)$.*

11.4 Leader Election and Clock Synchronization

The goal of this section is to show that any multi-hop wake-up protocol WakeUp (deterministic or randomized) can be transformed into a leader election protocol or a clock synchronization protocol, with only a logarithmic overhead. Consider a multi-hop wake-up protocol WakeUp, and denote by $T = T(n)$ its time complexity on any n-node radio network. We represent node ids as binary strings of length $\log n + 1$, and by $\mathrm{id}_v[i, .., j]$ we denote the string consisting of the bits of id_v on positions $i, i + 1, ..., j$, counting from left to right.

11.4.1 Leader Election Protocol

The leader election protocol, called Elect, is based on routine WakeUp solving the wake-up problem. It consists of two parts: *wake-up part* and *election part*. The election part consists of $\log n$ stages. Before giving a pseudocode of the protocol, we provide an informal description of both parts.

11.4.1.1 Wake-Up Part

The goal of this part is to wake up all nodes. Each node v, after being activated by a message or spontaneously, starts executing the wake-up protocol WakeUp. After completing this protocol, node v suspends itself for T rounds, in order to ensure that the execution of the wake-up protocol WakeUp in this part does not overlap with the election part that follows.

11.4.1.2 Election Part

In this part, all nodes in the network select a node $\mathbb{z}$ with the smallest id as the leader. The selection process proceeds in $\log n + 1$ *stages*, and the goal of stage i, for $i = 0, 1, \dots, \log n$, is to distribute the ith highest bit of $\mathrm{id}_{\mathbb{z}}$. Assume that at the beginning of stage $i > 0$ all nodes know the first i bits of $\mathrm{id}_{\mathbb{z}}$, and let *min_id_prefix* $= \mathrm{id}_{\mathbb{z}}[0, ..., i-1]$ be the string consisting of these bits; in the beginning of stage $i = 0$ we assume that this string is empty. Each node v for which $\mathrm{id}_v[0, ..., i-1] =$ *min_id_prefix* holds is a potential candidate for becoming a leader. If such a candidate node v has a 0 on its ith bit, it executes WakeUp routine in its stage i in order to inform other nodes about its existence. If at least one wake-up process is initiated in stage i, by the end of this stage all nodes conclude that the ith most significant bit in $\mathrm{id}_{\mathbb{z}}$ must be 0; otherwise this ith bit is set to 1.

Although we do not make any additional assumptions on the routine WakeUp used in the leader election protocol, except that it solves the wake-up problem, it needs to be executed in a specific way. Mainly, making a decision to execute WakeUp routine in a stage corresponds to the spontaneous wake-up of a node. Nodes that have not made such a decision in the current stage are treated as if they were asleep, i.e., they stay idle until they make such a decision in the current stage or till the end of the stage otherwise.

The main difficulty that needs to be overcome in the election part is the lack of synchronization. More precisely, we cannot guarantee that the executions of stage i, for $i = 0, 1, \dots, \log n$, in different nodes are synchronized in time. Instead, the only property we can guarantee is that the maximum offset between the activation times of any two nodes, and thus also between neighboring stages run in two different nodes, is at most T. In order to avoid simultaneous executions of routines WakeUp initiated in different stages, it is therefore sufficient to pad each stage with two waiting periods of length T each: one at the beginning and one at the end of the stage. More specifically, each stage is split into four sub-stages, each of length T. If a node v is a candidate for the leader and initiates the wake-up process by itself, this is done in the beginning of the second sub-stage. On the other hand, v can be activated to start its wake-up process by another node w at any time during the first, second, or third sub-stage. This is due to the offset, which is at most the length of a single sub-stage. This concludes the election part.

Below, we give a pseudocode of the protocol Elect executed at a node v. We denote by $\circ$ the concatenation operation on strings.

11.4.1.3 Analysis of Protocol Elect

We first argue about correctness. Let $\mathbb{z}$ be the node with the smallest id. After the wake-up part, the local clocks of nodes differ by at most T. This is because the activation of all nodes takes place during the execution of WakeUp routine by the first activated node, and this takes at most T rounds. This implies, in particular, that two stages of different nodes can overlap only if they are consecutive. Moreover, they can overlap on at most T rounds. More specifically, the only possible overlap is when the fourth sub-stage of a node in stage i overlaps the first sub-stage of another node that is in stage $i+1$. Consequently, there are no interferences between wake-up processes run in different stages in the election part. This property together with a straightforward inductive argument guarantees that there is no wake-up process in stage i iff the ith bit of the id of node $\mathbb{z}$ is 1. This concludes the proof of correctness.

Algorithm 2 Elect

```
Upon activation:
$clock \leftarrow 0$
Execute WakeUp (in $T$ rounds)
Wait until $clock = 2T$
$min\_id\_prefix \leftarrow [\,]$                    // empty string
For $r \leftarrow 0, 1, \ldots, \log n$ do
    $bit \leftarrow 1$                    // stage $i$ starts, local time clock is $(4i + 2)T$
    If $id_v[0, ..., i] = min\_id\_prefix \circ 0$ then
        $bit \leftarrow 0$
        Wait until a message received or $clock = (4i + 3)T$
    Else
        Wait until a message received or $clock = (4i + 5)T$
        If a message received then $bit \leftarrow 0$
    If $bit = 0$ then execute WakeUp (in $T$ rounds)
    $min\_id\_prefix \leftarrow min\_id\_prefix \circ bit$
    Wait until $clock = (4i + 6)T$
Output: $min\_id\_prefix$
```

The total number of rounds from the first activation until the last termination is at most $2T + 4T \cdot (\log n + 1) + T$, where the last T comes from the offset. Thus we obtained:

Theorem 8 (Chrobak et al. [20]) *Suppose that* WakeUp *is a wake-up protocol with running time $T(n)$. Then* WakeUp *can be converted into a leader election protocol* Elect *with running time $O(T(n) \log n)$.*

Taking deterministic wake-up protocol CK, instantiated as in Corollaries 2 and 3, or randomized protocol MIFS, as subroutine WakeUp, we get the following result:

Corollary 4 *Protocol* Elect *based on suitably instantiated wake-up protocol* CK *solves leader election in $O(\min\{n, D\Delta\} \log \Delta \log^2 n)$ rounds (existential version) or in $O(\min\{n, D\Delta\}\Delta \operatorname{polylog} n)$ rounds (explicit version).*

Protocol Elect *based on randomized wake-up protocol* MIFS *solves leader election in $O(D \log^2 n \log(n/\varepsilon))$ rounds with probability at least $1 - \varepsilon$.*

11.4.2 Clock Synchronization

It is relatively simple to extend the leader election protocol Elect, described in the previous section, to perform clock synchronization. We start by running the routine Elect. Once a leader $\mathbb{z}$ has been elected, node $\mathbb{z}$ waits for T rounds to allow all nodes to terminate their Elect executions. Next, node $\mathbb{z}$ broadcasts its local time to all other nodes using the same wake-up protocol WakeUp. During this execution, each round of the propagated value is increased by one. Upon activation, a node adopts the received value as its local clock setting and joins the execution of WakeUp.

Theorem 9 (Chrobak et al. [20]) *Let* WakeUp *be a wake-up protocol with running time* $T(n)$*. Then* WakeUp *can be converted into a clock synchronization protocol* ClockSynch *with running time* $O(T(n)\log n)$.

Applying efficient leader election protocol Elect as in Corollary 4, together with its corresponding wake-up subroutine WakeUp, we obtain the following result:

Corollary 5 *Protocol* ClockSynch*, based on suitably instantiated wake-up protocol* CK *and leader election protocol* Elect*, solves the local clock synchronization problem in* $O(\min\{n, D\Delta\}\log\Delta\log^2 n)$ *rounds (existential version) or in* $O(\min\{n, D\Delta\}\Delta\,\text{polylog}\,n)$ *rounds (explicit version).*

Protocol ClockSynch*, based on randomized wake-up protocol* MIFS *and leader election protocol* Elect *(with* MIFS *used as the wake-up subroutine), solves the local clock synchronization problem in* $O(D\log^2 n\log(n/\varepsilon))$ *rounds with probability at least* $1-\varepsilon$.

11.5 Mutual Exclusion

In this section we consider the mutual exclusion problem on the multiple-access channel. We assume slightly more restricted (and thus more realistic) model when no node can transmit and listen at the same time. We start with giving a definition of the problem, followed by case study of one-entry mutual exclusion and its "local" version, called ε-exclusion. Then we show how to extend one-entry solutions to general dynamic mutual exclusion or ε-exclusion algorithms, in particular, how to achieve fairness in potentially unbounded executions.

Mutual Exclusion—Problem Definition

A mutual exclusion is an access control kind of problem, where a concurrent program run by independent devices occasionally requests an access to the shared object. More precisely, each node executes a protocol partitioned into the following four sections:

Entry (trying) the part of the protocol executed in preparation for entering the critical section;
Critical the part of the protocol to be protected from concurrent execution;
Exit the part of the protocol executed on leaving the critical section;
Remainder the rest of the protocol.

These sections are executed cyclically in the order *remainder, entry, critical*, and *exit*. Intuitively, the remainder section corresponds to local computation of a node, the critical section corresponds to the access to the shared object (the channel in our case), though we abstract from particular purpose and operations done within each of these sections (it is not a part of the problem). The goal of the mutual exclusion problem is to develop entry and exit sections so that for any remainder and critical sections several properties of execution are guaranteed (they will be specified later).

In the traditional mutual exclusion problem, as defined in [4] in the context of shared-memory model, the adversary controls the remainder and critical sections. In particular, it controls their duration in each cycle, only subject to the obvious assumptions that this duration in each cycle is finite or the last performed section is the remainder one. The mutual exclusion algorithm, on the other hand, provides a protocol for the entry and exit sections of each node. In this sense, the mutual exclusion problem can be seen as a game between the adversary controlling the lengths of remainder and critical sections of each node (each such section for each node may have different lengths) and the algorithm controlling sections' entry and exit. The goal of the algorithm is to guarantee several useful properties of the execution (to be defined later), while the goal of the adversary is to fail them. Note that the sections controlled by the adversary and those controlled by the algorithm are interleaved in the execution. Additionally, in order to make the game fair, it is typically assumed that every variable used by the algorithm, i.e., in the entry and exit sections, cannot be accessed by the adversary in the critical and remainder sections.

We consider a multiple-access channel model of communication. In a single round, a node in the entry or the exit section can do the following: perform some action on the channel (either transmit a *single-bit* message or listen), do some local computation, and change its section either from entry to critical or from exit to remainder. We assume that changing sections occurs momentarily between consecutive rounds, i.e., in each round a node is exactly in one section of the protocol.

A multiple-access channel is both the (only) communication medium and the exclusively shared object. Therefore, additional restrictions, not used in other classic models (e.g., in shared memory), must be imposed to bind the use of the channel during the remainder and the critical sections:

- the channel must not be used by nodes being in the remainder section, i.e., such nodes neither listen nor transmit a message;
- a node in the critical section transmits a special message, called the *critical message*, in each round.

If some of these conditions were not satisfied, the adversary would have an unlimited power of creating collisions in the channel, thus preventing any communication.

A solution of the mutual exclusion problem should satisfy the following four properties:

Exclusion: in every round of any execution, at most one node is in the *critical* section.

No deadlock: in every round i of an execution, if there is a node in the entry section at round i, then *some node* will enter the *critical* section eventually after round i.

No lockout: in every round i of an execution, if a node p is in the entry section at round i, then *node* p will enter the *critical* section eventually after round i.

Obstruction freedom: in every round i of an execution, if a node p is in the exit section at round i, then node p will switch to the *remainder* section eventually after round i.

Note that the no-lockout property ensures, to some extent, fairness: each node that demands an access to the critical section will eventually get it. Observe that some mutual exclusion algorithms considered in the literature do not satisfy this property.

The exclusion condition is, as we will show, difficult to achieve in some cases, and when achieved, it is often for the price of large complexity. Therefore, we also consider a slightly weaker condition:

ε-Exclusion for every node p and for every critical section imposed by the adversary for p, the probability that in any round of this critical section there is another node being in the critical section is at most ε.

Intuitively, ε-exclusion guarantees mutual exclusion "locally," i.e., for every single execution of the critical section by a node and with probability at least $1 - \varepsilon$.

Complexity Measure

The basic measure of complexity is the maximum number of rounds in any interval in which there is some node in the entry section and there is no node in the critical section. This measure is called *makespan*. Observe that an upper bound on makespan automatically implies no-deadlock property, but not necessarily no-lockout.

The definition of makespan requires more detailed specification in case of randomized solutions. We define a strategy $\mathcal{P}$ of the adversary as a set of n sequences, where each sequence corresponds to a different node and contains, subsequently interleaved, the lengths of remainder and critical sections of the corresponding node. It can be assumed that each sequence is either infinite or of even length; the meaning of the sequence of even length is that the last remainder section of the corresponding node lasts forever. For a given mutual exclusion algorithm $\mathcal{B}$ and adversarial strategy $\mathcal{P}$, let $\mathcal{M}(\mathcal{B}, \mathcal{P})$ be a random variable equal to the maximum length of a time interval in which there is some node in the entry section and there is no node in the critical section in an execution of algorithm $\mathcal{B}$ run against strategy $\mathcal{P}$. The maximum of expected values of variable $\mathcal{M}(\mathcal{B}, \mathcal{P})$, taken over all adversarial strategies $\mathcal{P}$, is the *expected makespan* of randomized algorithm $\mathcal{B}$. In case of the ε-mutual exclusion solutions, makespan is defined only for executions where the mutual exclusion property is always fulfilled, i.e., where there is no round with at least two nodes being in the critical section.

11.5.1 From Wake-Up to Mutual Exclusion

We start with quoting a $\Omega(n)$ lower bound on expected makespan shown in [9].

Theorem 10 (Bienkowshi et al. [9]) *Expected makespan of any randomized mutual exclusion algorithm is at least $n/2$ in the absence of collision detection capability, even in the setting with global clock and with knowledge of the number n of nodes.*

11.5.1.1 Deterministic One-Entry Mutual Exclusion Algorithm

Universal synchronizers, or even basic synchronizers, can be used to obtain solution for a simpler version of mutual exclusion, called *one-entry mutual exclusion*, which is a form of (one-instance) consensus problem [25]. In this problem the aim is to make one of the woken-up nodes (i.e., being in entry section) to enter the critical section and to possess the channel, while others must remain silent (i.e., remain in the entry section).

Assume we are given a (n, n)-synchronizer of length m. Note that the main challenge is to use the synchronizer in the model where a node cannot transmit and listen simultaneously in a single round. In particular, a successful transmission is heard only by other active nodes, while the transmitter may not be aware whether its transmission was successful or not. This problem can be solved by the following deterministic algorithm.

Algorithm 3 KnownNumber

Upon entering critical section, node v listens for m rounds. If it hears some node id in one of these rounds, it "decides" on this id to be the winner of the channel and remains silent forever. If it hears silence in all these m rounds, it starts transmitting its id according to the schedule defined by the corresponding row of the synchronizer. If it hears some node id in one of the following m rounds (in some round when it is listening), it decides on this node to be the "winner" and remains silent forever. If it does not hear any message in all the $2m$ rounds, it decides to enter the critical section and transmits the message "my id v won the channel" forever.

Theorem 11 (Czyzowicz et al. [25]) *Algorithm* KnownNumber *solves one-entry mutual exclusion in time $O(n \log^2 n)$ after first node enters its entry section, for any known number n of nodes.*

Obviously the above theorem guarantees only exclusion and no-deadlock properties, since other two properties are specific for dynamic version of mutual exclusion.

11.5.1.2 From One-Entry to Dynamic Mutual Exclusion

We present a generic mutual exclusion algorithm, called MacMEx, which uses a one-entry algorithm as a subroutine and solves the general problem of dynamic mutual exclusion, preserving the complexity of the one-entry solution.

First observe that algorithm KnownNumber has the following two properties, crucial from the point of view of extension to dynamic mutual exclusion:

P1. Every node listens in the round in which it enters the critical section.

P2. Starting from the round in which node v decides to enter the critical section, node v transmits the message "my id v won the channel" forever, and all other nodes listen forever.

We present an algorithm which, given a one-entry mutual exclusion subroutine satisfying properties P1 and P2, guarantees exclusion, no-deadlock, and obstruction freedom in dynamic mutual exclusion.

Algorithm 4 MacMEx

Entry section. Node i executes a given one-entry subroutine satisfying properties P1 and P2, until one of the following events occurs:

- node v decides to enter the critical section;
  in this case node v enters the critical section
- node v hears either the message "occupied" or the message "my id w won the channel";
  in this case node v stops the execution of the one-entry subroutine and keep listening on the channel in the next rounds until the next case occurs
- node v hears the message "released";
  in this case node v starts a new execution of the one-entry subroutine

Critical section. Node v transmits the message "occupied" on the channel in each round when it is in the critical section. The rest of the behavior of the node in this section is controlled by the adversary.

Exit section. Node v transmits the message "released" on the channel and leaves the exit section.

The proof of correctness of Algorithm MacMEx, including analysis of exclusion, no-deadlock, and obstruction freedom properties, is based on the following invariant.

Lemma 5 (Czyzowicz et al. [25]) *Exactly one of the following properties holds in any round t:*

Q1 the message "occupied" is heard in round t and its sender is the only node in the critical section in this round; additionally, no node is in the exit section and no node executes the one-entry subroutine in round t; or
Q2 the message "released" is heard in round t and its sender is the only node in the exit section in this round; additionally, no node is in the critical section and no node executes the one-entry subroutine in round t; or
Q3 there is at least one node executing the one-entry subroutine in round t; additionally, all such nodes are exactly those in the entry section and no node is in the critical or exit sections in round t; or
Q4 all nodes are in the remainder section in round t.

Using Lemma 5, the following result can be proved, by analyzing how cases Q1–Q4 interleave in the execution of algorithm MacMEx.

Theorem 12 (Czyzowicz et al. [25]) *Algorithm* MacMEx *with a one-entry subroutine satisfying properties P1 and P2 is a mutual exclusion algorithm with* *no-deadlock and no obstruction. Moreover, the makespan of the* MacMEx *algorithm is the same as the time complexity of the one-entry subroutine.*

Combining Theorems 12 and 11 for the one-entry version of the problem, we derive the following conclusions for dynamic mutual exclusion.

Theorem 13 (Czyzowicz et al. [25]) *Algorithm* MacMEx *is a dynamic mutual exclusion algorithm with no-deadlock and no obstruction in a multiple-access channel, with makespan* $O(n \log^2 n)$.

11.5.1.3 Randomized ε-Exclusion Algorithm

In this scenario, we build our solution based on the IncreaseFromSquare algorithm of [38], studied in Sect. 11.2.2. We describe how to extend it to meet the requirements of ε-exclusion. It is enough to specify a one-entry mutual exclusion version of the algorithm satisfying properties P1 and P2 and then apply a generic scheme analogous to the deterministic MacMEx protocol. An idea of the one-entry extension is similar to the use of synchronizers in the deterministic algorithm KnownNumber. When a node enters the entry section, it first goes into the listening mode and stays in this mode for $T = O(\log n \cdot \log(1/\varepsilon))$ rounds, where T is the time complexity of algorithm IncreaseFromSquare. If within this time the node hears a message from another node, it stops. Afterward, the node starts to execute the IncreaseFromSquare algorithm. Whenever it is not transmitting, it listens, and when it hears a message from other nodes it stops. After executing T rounds of the subroutine IncreaseFromSquare, the node enters the critical section. The following result holds:

Theorem 14 (Bienkowski et al. [9]) *The extended version of the* Increase-FromSquare *algorithm solves the* ε*-exclusion problem with makespan* $O(\log n \cdot \log(1/\varepsilon))$.

11.5.1.4 Fairness

The mutual exclusion algorithms presented so far do not guarantee the *no-lockout* property, i.e., it may happen that a node never gets out of its entry section, as other nodes exchange access to the critical section among themselves. In this section we show how to modify algorithms satisfying only exclusion, no-deadlock, and obstruction freedom properties in such a way that the additional no-lockout property is also fulfilled.

The modification is as follows. Each node keeps an additional local counter of losses. When it starts its entry section, it sets its counter to zero and whenever it loses the competition for the critical section, it increases this counter by one (it happens whenever another node enters the critical section). When a node enters its exit section, it becomes a guard: it will help the nodes currently being in their entry section to choose one of them with the highest counter of losses. How high the losses counter can grow is bounded by the number of nodes being in their entry sections at the moment when the considered node entered its entry section. This implies that the time after which the node will enter the critical section is finite.

More details follow. When a node is in the critical section, it still broadcasts the message "occupied." When it enters its exit section, it broadcasts the "release" message. The node still runs its exit section in three-bit/round blocks, where the first two bits/rounds are broadcasts of 1 and 0 by the guard only; the guard broadcasts a 1 also in the third round of each block in order to simulate collision detection for the competitors. Simultaneously, after hearing the "release" message on the channel, nodes awaiting in their entry section start computation as well; they are now competitors for the next successful entry to the critical section. In the ith bit of the first $\lceil \log n \rceil$ blocks, the competitors send a 1 if their ith highest bit is 1. If a competitor is listening in the third round of a block and it does not hear a 1, it switches off, as it has lost (the heard 1 comes from the guard, silence means collision and thus another competitor having a 1 on a higher bit position). After this phase, the guard continues its exit section for additional $\lceil \log n \rceil$ blocks, and only the competitors with the highest number of losses continue their entry section during this period. They use their ids instead of the losses counters, in order to choose exactly one of them. After these $2\lceil \log n \rceil$ blocks, the guard leaves its exit section. If there were any competitors in the beginning of this phase, exactly one of them survived and it enters the critical section. If not, the channel is free; in particular, the nodes which have entered the critical section in the meantime, if any, will start competing for the channel after the guard leaves its exit section (as they have their losses counter equal to 0).

Lemma 6 (Bienkowski et al. [9]) *The described modification guarantees the no-lockout property for any mutual exclusion algorithm, at the cost of slowing down the original algorithm by a constant factor and with an additional cost of $O(\log n)$ rounds.*

Combining Lemma 6 with Theorems 13 and 14, we get the following results:

Corollary 6 *There exists a deterministic algorithm solving the mutual exclusion problem with no-lockout and with makespan $O(n \log^2 n)$.*

Corollary 7 *There exists an algorithm solving the ε-exclusion problem with no-lockout and with makespan $O(\log n \cdot \log(1/\varepsilon))$.*

11.6 Remarks and Open Problems

Several interesting research problems remain open. In one-instance coordination problems, there is a natural question about closing the (poly-)logarithmic gaps between upper and lower bounds. Moreover, there is an issue of efficient constructions of more sophisticated synchronizers, which can be later used for developing more efficient and more practical solutions for several dynamic control problems, such as dynamic broadcast or dynamic mutual exclusion. In case of multi-hop radio networks, there is a dearth of solutions for dynamic coordination and communication problems.

The second group of problems consists of efficient partially synchronous communication. This problem was stated by Pelc [48]. It was also considered in the context of centralized radio communication by Chlebus and Rokicki [18]. A related issue of non-deterministic feedback and failures (cf. [19, 23]) and mobility also becomes important from the point of view of coordination and control problems.

In this work we considered solutions in the context of radio networks without access to the global clock and without collision detection, while still having disjoint identifiers from some known range. Other realistic settings can be considered in the context of coordination problems in wireless networks, including additional knowledge or capabilities, more restricted model, multi-frequency communication [26], different methods of decoding a transmitted signal (e.g., SINR [33], transmission-interference model), and specific topologies (e.g., geometric graphs).

Acknowledgments This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/G023018/1].

References

1. N. Abramson. Development of the alohanet, *IEEE Transactions on Information Theory* 31:119–123, (1985).
2. N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast, *Journal of Computer and System Sciences* 43:290–298, 1991.
3. G. Alonso, E. Kranakis, C. Sawchuk, R. Wattenhofer, and P. Widmayer. Randomized protocols for node discovery in ad-hoc multichannel broadcast networks, In: *Proceedings, 2nd Annual Conference on Adhoc Networks and Wireless (ADHOCNOW)*, volume 2865 of *Lecture Notes in Computer Science*. Springer, pages 104–115, 2003.
4. H. Attiya and J. Welch. *Distributed Computing*, Wiley, 2004.
5. E. Bach and J. Shallit. *Algorithmic Number Theory*, Volume I, The MIT Press, Cambridge, MA, 1996.
6. R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time complexity of broadcast in radio networks: An exponential gap between determinism and randomization, *Journal of Computer and System Sciences*, 45:104–126, 1992.
7. R. Bar-Yehuda, A. Israeli, and A. Itai. Multiple communication in multihop radio networks, *SIAM Journal on Computing*, 22:875–887, 1993.
8. M.A. Bender, M. Farach-Colton, S. He, B.C. Kuszmaul, and C.E. Leiserson. Adversarial contention resolution for simple channels, In: *Proceedings, 17th Annual ACM Symposium on Parallel Algorithms (SPAA)*, pages 325–332, 2005.
9. M. Bienkowski, M. Klonowski, M. Korzeniowski, and D.R. Kowalski. Dynamic sharing of a multiple access channel, In: *Proceedings, 27th International Symposium on Theoretical Aspects of Computer Science (STACS)*, LIPIcs 5 Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, pages 83–94, to appear, 2010.
10. A. Borodin, J.M. Kleinberg, P. Raghavan, M. Sudan, and D.P. Williamson. Adversarial queuing theory, *Journal of the ACM*, 48:13–38, 2001.
11. J. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Transactions on Information Theory*, 25:505–515, 1979.
12. I. Chlamtac and S. Kutten. On broadcasting in radio networks - problem analysis and protocol design, *IEEE Transactions on Communications*, 33:1240–1246, 1985.

13. B.S. Chlebus, L. Gasieniec, A. Gibbons, A. Pelc, and W. Rytter. Deterministic broadcasting in unknown radio networks, *Distributed Computing*, 15:27–38, Springer, 2002.
14. B.S. Chlebus, L. Gasieniec, D. Kowalski, and T. Radzik. On the wake-up problem in radio networks, In: *Proceedings, 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3580 of *Lecture Notes in Computer Science*. Springer, Lisbon, Portugal, pages 347–359, 2005.
15. B.S. Chlebus and D. Kowalski. A better wake-up in radio networks, In: *Proceedings, 23rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 266–274, 2004.
16. B.S. Chlebus, D.R. Kowalski, and T. Radzik. Many-to-many communication in radio networks, *Algorithmica*, 54:118–139, 2009.
17. B.S. Chlebus, D.R. Kowalski, and M.A. Rokicki. Adversarial queuing on the multiple-access channel, In: *Proceedings, 25th ACM Symposium on Principles of Distributed Computing (PODC)*, ACM, Denver, USA, pages 92–101, 2006.
18. B.S. Chlebus and M.A. Rokicki, Centralized asynchronous broadcast in radio networks, *Theoretical Computer Science*, 383:5–22, 2007.
19. G. Chockler, M. Demirbas, S. Gilbert, N.A. Lynch, C.C. Newport, and T. Nolte. Consensus and collision detectors in radio networks, *Distributed Computing*, 21:55–84, 2008.
20. M. Chrobak, L. Gasieniec, and D.R. Kowalski. The wake-up problem in multihop radio networks, *SIAM Journal on Computing*, 36:1453–1471, 2007.
21. M. Chrobak, L. Gasieniec, and W. Rytter. Fast broadcasting and gossiping in radio networks, *Journal of Algorithms*, 43:177–189, 2002.
22. A.E.F. Clementi, A. Monti, and R. Silvestri. Distributed broadcast in radio networks of unknown topology, *Theoretical Computer Science*, 302:337–364, 2003.
23. A.E.F. Clementi, A. Monti, and R. Silvestri. Round robin is optimal for fault-tolerant broadcasting on wireless networks, In: *Proceedings, 9th Annual European Symposium on Algorithms (ESA)*, volume 2161 of *Lecture Notes in Computer Science*. Springer, Aarhus, Denmark, pages 452–463, 2001.
24. A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology, In: *Proceedings, 44th IEEE Symposium on Foundations of Computer Science (FOCS)*, EEE Computer Society, Cambridge, USA, pages 492–501, 2003.
25. J. Czyzowicz, L. Gasieniec, D.R. Kowalski, and A. Pelc. Consensus and mutual exclusion in a multiple access channel, In: *Proceedings, (DISC)*, volume 5805 of *Lecture Notes in Computer Science*. Springer, Elche, Spain, pages 512–526, 2009.
26. S. Dolev, S. Gilbert, R. Guerraoui, F. Kuhn, and C.C. Newport. The wireless synchronization problem, In: *Proceedings, (PODC)*, ACM, Calgary, Canada, pages 190–199, 2009.
27. S. Dolev, S. Gilbert, R. Guerraoui, and C.C. Newport. Gossiping in a multi-channel radio network, In: *Proceedings, (DISC)*, volume 4731 of *Lecture Notes in Computer Science*. Springer, Lemesos, Cyprus, pages 208–222, 2007.
28. M. Farach-Colton and M.A. Mosteiro. Initializing sensor networks of non-uniform density in the Weak Sensor Model, In: *Proceedings, 10th International Workshop on Algorithms and Data Structures (WADS)*, volume 4619 of *Lecture Notes in Computer Science*. Springer, Halifax, Canada, pages 565–576, 2007.
29. R.G. Gallager. A perspective on multiaccess channels, *IEEE Transactions on Information Theory*, 31:124–142, 1985.
30. L. Gasieniec, A. Pelc, and D. Peleg. The wakeup problem in synchronous broadcast systems, *SIAM Journal on Discrete Mathematics*, 14:207–222, 2001.
31. A.G. Greenberg and S. Winograd. A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels, *Journal of ACM* 32:589–596, 1985.
32. S. Gilbert, R. Guerraoui, and C.C. Newport. Of malicious motes and suspicious sensors: On the efficiency of malicious interference in wireless networks, *Theoretical Computer Science*, 410:546–569, 2009.
33. O. Goussevskaia, Y.A. Oswald, and R. Wattenhofer. Complexity in geometric SINR, In: *Proceedings, (MobiHoc)*, ACM, Montreal, Canada, pages 100–109, 2007.

34. J.F. Hayes. An adaptive technique for local distribution, *IEEE Transactions on Communications*, 26:1178–1186, 1978.
35. J. Hrastad, T. Leighton, and B. Rogoff. Analysis of backoff protocols for multiple access channels, *SIAM Journal on Computing*, 25:740–774, 1996.
36. P. Indyk. Explicit constructions of selectors and related combinatorial structures, with applications, In: *Proceedings, 13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, ACM/SIAM, San Francisco, USA, pages 697–704, 2002.
37. T. Jurdzinski, M. Kutylowski, and J. Zatopianski. Efficient algorithms for leader election in radio networks, In: *Proceedings, 21st Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 51–57, 2002.
38. T. Jurdzinski and G. Stachowiak. Probabilistic algorithms for the wakeup problem in single-hop radio networks, In: *Proceedings, 13th International Symposium on Algorithms and Computation (ISAAC)*, volume 2518 of *Lecture Notes in Computer Science*. Springer, Vancouver, Canada, pages 535–549, 2002.
39. W.H. Kautz and R.R.C. Singleton. Nonrandom binary superimposed codes, *IEEE Transactions on Information Theory*, 10:363–377, 1964.
40. C.-Y. Koo, V. Bhandari, J. Katz, and N.H. Vaidya. Reliable broadcast in radio networks: The bounded collision case, In: *Proceedings, 25th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, ACM, Denver, USA, pages 258–264, 2006.
41. D. Kowalski, On selection problem in radio networks, In: *Proceedings, 24th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 158–166, 2005.
42. D.R. Kowalski and A. Pelc. Faster deterministic broadcasting in ad hoc radio networks, *SIAM Journal on Discrete Mathematics* 18:332–346, 2004.
43. D.R. Kowalski and A. Pelc. Deterministic broadcasting time in radio networks of unknown topology, Broadcasting in undirected ad hoc radio networks, *Distributed Computing* 18:43–57, Springer, 2005.
44. D.R. Kowalski and A. Pelc. Leader election in ad hoc radio networks: A keen ear helps, In: *Proceedings, 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5556 of *Lecture Notes in Computer Science*. Springer, Rhodes, Greece, pages 521–533, 2009.
45. E. Kushilevitz and Y. Mansour. An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks, *SIAM Journal on Computing*, 27:702–712, 1998.
46. R.M. Metcalfe and D.R. Boggs. Ethernet: Distributed packet switching for local computer networks, *Communications of the ACM*, 19:395–404, 1976.
47. K. Nakano, S. Olariu. Uniform leader election protocols for radio networks, *IEEE Transactions on Parallel Distributed Systems*, 13:516–526, 2002.
48. A. Pelc. Activating anonymous ad hoc radio networks, *Distributed Computing*, 19:361–371, 2007.
49. A. Pelc and D. Peleg. Feasibility and complexity of broadcasting with random transmission failures, In: *Proceedings, 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, ACM, Las Vegas, USA, pages 334–341, 2005.
50. D. Peleg. Time-efficient broadcasting in radio networks: A review, In: *Proceedings, 4th International Conference on Distributed Computing and Internet Technology (ICDCIT)*, volume 4882 of *Lecture Notes in Computer Science*. Springer, Bangalore, India, pages 1–18, 2007.
51. R. Rom and M. Sidi. *Multiple Access Protocols: Performance and Analysis*, Springer, New York, NY, 1990.
52. A. Ta-Shma, C. Umans, and D. Zuckerman. Loss-less condensers, unbalanced expanders, and extractors, In: *Proceedings, 33rd ACM Symposium on Theory of Computing (STOC)*, ACM, Heraklion, Crete, Greece, pages 143–152, 2001.
53. B.S. Tsybakov and V.A. Mikhailov. Free synchronous packet access in a broadcast channel with feedback, *Prob Inf Transmission* 14:259–280, 1978. (Translated from Russian original in *Problemy Peredachi Informatsii*, 1977.)
54. D.E. Willard. Log-logarithmic selection resolution protocols in a multiple access channel, *SIAM Journal on Computing*, 15:468–477, 1986.

Part IV
Data Propagation and Collection

Chapter 12
Probabilistic Data Propagation in Wireless Sensor Networks

Sotiris Nikoletseas and Paul G. Spirakis

Abstract We study the problem of data propagation in distributed wireless sensor networks. We present two characteristic methods for data propagation: the first one performs a local, greedy optimization to minimize the number of data transmissions needed, while the second creates probabilistically optimized redundant data transmissions to trade off energy efficiency with fault tolerance. Both approaches make use of randomization at both the algorithmic design and analysis; this demonstrates the suitability for distributed sensor network algorithms of probabilistic techniques, because of their simplicity, locality, efficiency, and load-balancing features.

12.1 Introduction

12.1.1 A Brief Overview of Wireless Sensor Networks

Recent dramatic developments in micro-electro-mechanical (MEMS) systems, wireless communications, and digital electronics have already led to the development of small in size, low-power, low-cost sensor devices. Such extremely small devices integrate sensing, data processing, and wireless communication capabilities. Also, they are equipped with a small but effective operating system and are able to switch between "sleeping" and "awake" modes to save energy. Pioneering groups (like the "Smart Dust" Project at Berkeley, the "Wireless Integrated Network Sensors" Project at UCLA, and the "Ultra low Wireless Sensor" Project at MIT) pursue diverse important goals, like a total volume of a few cubic millimeters and extremely low energy consumption, by using alternative technologies, based on radio frequency (RF) or optical (laser) transmission.

Examining each such device individually might appear to have small utility, however, the effective *distributed co-ordination* of large numbers of such devices can lead to the efficient accomplishment of large sensing tasks. Large numbers of sensor nodes can be deployed (usually in an ad hoc manner) in areas of interest

S. Nikoletseas (✉)
Research Academic Computer Technology Institute (RACTI) and Department of Computer Engineering and Informatics, University of Patras, Patras, Greece
e-mail: nikole@cti.gr

(such as inaccessible terrains, disaster locations, and ambient intelligence settings) and use *self-organization and collaborative methods* to spontaneously form a sensor network.

Their wide range of applications is based on the possible use of various sensor types (i.e., thermal, visual, seismic, acoustic, radar, magnetic) in order to monitor a wide variety of conditions (e.g., temperature, object presence and motion, humidity, pressure, noise levels). Thus, sensor networks can be used for continuous sensing, reactive event detection, location sensing as well as micro-sensing. Hence, sensor networks have important applications, including (a) environmental applications (such as fire detection, flood detection, precision agriculture), (b) health applications (like telemonitoring of human physiological data), (c) security applications (nuclear, biological, and chemical attack detection), and (d) home applications (e.g., smart environments and home automation). For an excellent survey of wireless sensor networks see [1] and also [7, 14].

We note that a single, universal, and technology independent model is missing in the state of the art, so in each protocol we present we precisely state the particular modeling assumptions (weaker or stronger) needed. Also, in Sect. 12.1.3 we make an attempt to define a hierarchy of models and relations between them.

12.1.2 Critical Challenges

The efficient and robust realization of such large, highly dynamic, complex, non-conventional networking environments is *a challenging algorithmic and technological task*. Features including the huge number of sensor devices involved, the severe power, computational and memory limitations, their dense deployment and frequent failures, pose *new design, analysis, and implementation aspects* which are essentially different not only with respect to distributed computing and systems approaches but also to ad hoc (such as mobile or radio) networking techniques.

We emphasize the following characteristic differences between sensor networks and ad hoc networks:

- The number of sensor particles in a sensor network is extremely large compared to that in a typical ad hoc network.
- Sensor networks are typically prone to frequent faults.
- Because of faults as well as energy limitations, sensor nodes may (permanently or temporarily) join or leave the network. This leads to highly dynamic network topology changes.
- The density of deployed devices in sensor networks is much higher than in ad hoc networks.
- The limitations in energy, computational power, and memory are much more severe.

Because of the above rather unique characteristics of sensor networks, efficient and robust distributed protocols and algorithms should exhibit the following critical properties:

12.1.2.1 Scalability

Distributed protocols for sensor networks should be highly scalable, in the sense that they should operate efficiently in extremely large networks composed of huge numbers of nodes. This feature calls for an urgent need to prove by analytical means and also validate (by large scale simulations and experimental testbeds) certain efficiency and robustness (and their trade-offs) guarantees for asymptotic network sizes. A protocol may work well for a few sensors, but its performance (or even correctness) may dramatically deteriorate as the deployment scale increases.

12.1.2.2 Efficiency

Because of the severe energy limitations of sensor networks and also because of their time-critical application scenarios, protocols for sensor networks should be very efficient, with respect to both energy and time.

12.1.2.3 Fault Tolerance

Sensor particles are prone to several types of faults and unavailability and may become inoperative (permanently or temporarily). Various reasons for such faults include physical damage during either the deployment or the operation phase, permanent (or temporary) cease of operation in the case of power exhaustion (or because of energy saving schemes, respectively). The sensor network should be able to continue its proper operation for as long as possible despite the fact that certain nodes in it may fail. Self-stabilization of the protocol is an aspect of fault tolerance which is especially relevant in this network setting.

12.1.3 Models and Relations Between Them

We note that a single, universal, technology- and cost-independent abstract model is missing in the state of the art, so for each protocol and problem we precisely state the particular modeling assumptions (weaker or stronger) needed. Also, we provide the following hierarchy of models and relations between them. Clearly this is just a starting point and a partial contribution to this matter of great importance.

12.1.3.1 Basic Model M_0

A *sensor cloud* (a set of sensors, which may be called grain particles) is spread in a region of interest (for a graphical presentation, see Fig. 12.1). The deployment is random uniform. Two important network parameters (crucially affecting topology and connectivity) are the particle density d (usually measured in numbers of $particles/m^2$) and the maximum transmission range $\mathcal{R}$ of each grain particle. The sensors do not move. A two-dimensional framework is adopted.

There is a single point in the network, which is called the “sink” S, that represents the fixed control center where data should be reported to. The sink is very powerful,

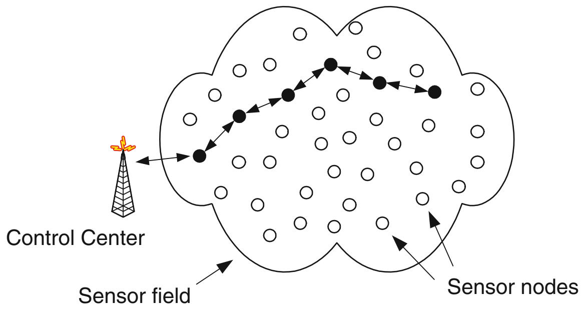


Fig. 12.1 A sensor cloud

in terms of both energy and computing power. Sensors can be aware of the direction (and position) of the sink, as well as of their distance to it. Such information can be, e.g., obtained during a setup phase, by having the (powerful) sink broadcast tiny control messages to the entire network region.

We assume that events, that should be reported to the control center, occur in the network region. Furthermore, we assume that these events are happening at random uniform positions.

As far as energy dissipation is concerned, we basically assume that the energy spent at a sensor when transmitting data is proportional to the square of the transmitting distance. Note, however, that the energy dissipation for receiving is not always negligible with respect to the energy when sending such as in case when transmissions are too short and/or radio electronics energy is high (see [10]).

A variation of this basic model includes multiple and/or mobile sinks. The following "wall" notion generalizes that of the sink and may correspond to multiple (and/or moving) sinks. A *receiving wall* $\mathcal{W}$ is defined to be an infinite (or at least quite large) line in the sensor cloud plane. Any particle transmission within range $\mathcal{R}$ from the wall $\mathcal{W}$ is received by $\mathcal{W}$. $\mathcal{W}$ is assumed to have very strong computing power, is able to collect and analyze received data, and has a constant power supply, and so it has no energy constraints. The wall represents in fact the authorities (the fixed control center) which the realization of a crucial event and collected data should be reported to. Note that a wall of appropriately big (finite) length may suffice. The notion of multiple sinks which may be static or moving has also been studied in [19], where Triantafilloy, Ntarmos, Nikoletseas, and Spirakis introduce "NanoPeer Worlds," merging notions from Peer-to-Peer Computing and Sensor Clouds.

The M_0 model is assumed for the LTP protocol.

12.1.3.2 A Stronger Model M_1

This model is derived by augmenting the model M_0 with additional (stronger) assumptions about the abilities of the sensors and the sensor deployment. In particular, the network topology can be a lattice (or grid) deployment of sensors. This structured placement of grain particles is motivated by certain applications, where it is possible to have a pre-deployed sensor network, where sensors are put (possibly by a human or a robot) in a way that they form a *2-dimensional lattice*. Note indeed that such sensor networks, deployed in a structured way, might be useful, e.g., in precise agriculture or smart buildings, where humans or robots may want to deploy the sensors in a lattice structure to monitor in a rather homogenous and uniform way certain conditions in the spatial area of interest. Also, exact terrain monitoring in military applications may also need some sort of a grid-like shaped sensor network. Note that Akyildiz et al. in a fundamental state of the art survey [1] refer to the pre-deployment possibility. Also, Intanagonwiwat et al. [11] explicitly refers to the lattice case. Moreover, as the authors of [11] state in an extended version of their work [12], they consider, for reasons of "analytic tractability," a square grid topology.

We further assume that each grain particle has the following abilities:

(i) It can estimate the direction of a received transmission (e.g., via the technology of direction-sensing antennae). We note that at least a gross sense of direction may be technologically possible by smart and/or multiple antennas. Regarding the extra cost of the antennas and circuitry, we note that certainly such a cost is introduced but as technology advances this cost may lower.
(ii) It can estimate the distance from a nearby particle that did the transmission (e.g., via estimation of the attenuation of the received signal).
(iii) It knows the direction toward the sink S. This can be implemented during a setup phase, where the (very powerful in energy) sink broadcasts the information about itself to all particles.
(iv) All particles have a common co-ordinates system.

Notice that GPS information is not assumed by this model. Also, the global structure of the network is not assumed known.

This model is used in the PFR protocol.

12.1.4 The Energy Efficiency Challenge in Routing

Since one of the most severe limitations of sensor devices is their limited energy supply (battery), one of the most crucial goals in designing efficient protocols for wireless sensor networks is minimizing the energy consumption in the network. This goal has various aspects, including: (a) minimizing the total energy spent in the network, (b) minimizing the number (or the range) of data transmissions, (c) combining energy efficiency and fault tolerance, by allowing redundant data transmissions which, however, should be optimized to not spend too much energy, (d) maximizing the number of "alive" particles over time, thus prolonging the system's lifetime, and (e) balancing the energy dissipation among the sensors in the network, in order to avoid the early depletion of certain sensors and thus the breakdown of the network (see, e.g., [8]). Even energy aspects related to potentially dangerous for health radiation levels (and their auto balancing) are relevant.

We note that it is very difficult to achieve all the above goals at the same time. There even exist trade-offs between some of the goals above. Furthermore, the importance and priority of each of these goals may depend on the particular application domain. Thus, it is important to have a variety of protocols, each of which may possibly focus at some of the energy efficiency goals above (while still performing well with respect to the rest goals as well).

In this chapter, we focus on online aspects of energy efficiency related to data propagation. In particular, for routing we present two methodologically characteristic energy efficient protocols:

- *The Local Target Protocol (LTP)* that performs a local optimization trying to minimize the number of data transmissions.
- *The Probabilistic Forwarding Protocol (PFR)* that creates redundant data transmissions that are probabilistically optimized, to trade off energy efficiency with fault tolerance.

Because of the complex nature of a sensor network (that integrates various aspects of communication and computing), protocols, algorithmic solutions, and design schemes for all layers of the networking infrastructure are needed. Far from being exhaustive, we mention the need for frequency management solutions at the physical layer and Medium Access Control (MAC) protocols to cope with multi-hop transmissions at the data link layer. Our approach focuses on routing and does not directly consider other important issues like link fading and hidden station interference, i.e., we assume that other network layers are resolving these issues. Also, regarding reliability, an endemic issue to wireless sensor networks, we consider permanent physical failures, i.e., we do not deal with bad values from physical sensing or transient failures. The interested reader may use the excellent survey by Akyildiz et al. [2] for a detailed discussion of design aspects and state of the art of all layers of the networking infrastructure.

12.2 LTP: A Single-Path Data Propagation Protocol

The LTP Protocol was introduced in [3]. The authors assume the basic model M_0 defined in Sect. 12.1.3.

12.2.1 The Protocol

Let $d(p_i,\ p_j)$ be the distance (along the corresponding vertical lines toward $\mathcal{W}$) of particles $p_i,\ p_j$ and $d(p_i, \mathcal{W})$ the (vertical) distance of p_i from $\mathcal{W}$. Let *info($\mathcal{E}$)* the information about the realization of the crucial event $\mathcal{E}$ to be propagated. Let p the particle sensing the event and starting the execution of the protocol. In this protocol, each particle p' that has received *info($\mathcal{E}$)* does the following:

- *Search phase*: It uses a periodic low-energy directional broadcast in order to discover a particle nearer to $\mathcal{W}$ than itself (i.e., a particle p'' where $d(p'', \mathcal{W}) < d(p', \mathcal{W})$).
- *Direct transmission phase*: Then, p' sends *info($\mathcal{E}$)* to p''.
- *Backtrack phase*: If consecutive repetitions of the *search phase* fail to discover a particle nearer to $\mathcal{W}$, then p' sends *info($\mathcal{E}$)* to the particle that it originally received the information from.

Note that one can estimate an a priori upper bound on the number of repetitions of the search phase needed, by calculating the probability of success of each search phase, as a function of various parameters (such as density, search angle, transmission range). This bound can be used to decide when to backtrack.

For a graphical representation see Figs. 12.2 and 12.3.

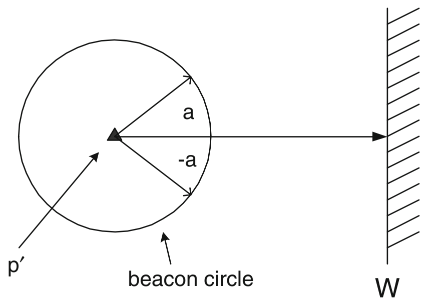


Fig. 12.2 Example of the search phase

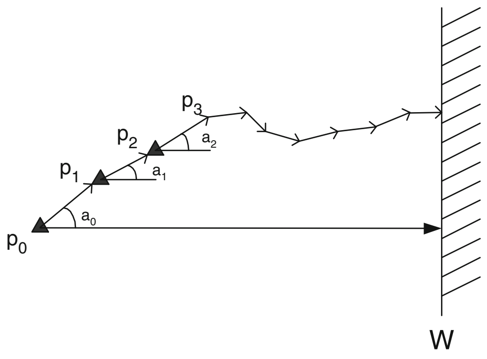


Fig. 12.3 Example of a transmission

12.2.2 Analysis of the Expected Hops Efficiency

We first provide some basic definitions.

Definition 1 Let $h_{\text{opt}}(p, \mathcal{W})$ be the (optimal) number of "hops" (direct, vertical to $\mathcal{W}$ transmissions) needed to reach the wall, in the *ideal* case when particles always exist in pairwise distances $\mathcal{R}$ on the vertical line from p to $\mathcal{W}$. Let Π be *a smart-dust propagation protocol*, using *a transmission path* of length $L(\Pi, p, \mathcal{W})$ to send info about event $\mathcal{E}$ to wall $\mathcal{W}$. Let $h(\Pi, p, \mathcal{W})$ be the actual number of hops (transmissions) taken to reach $\mathcal{W}$. The *"hops" efficiency* of protocol Π is the ratio

$$C_h = \frac{h(\Pi, p, \mathcal{W})}{h_{\text{opt}}(p, \mathcal{W})}$$

Clearly, the number of hops (transmissions) needed characterizes the (order of the) energy consumption and the time needed to propagate the information $\mathcal{E}$ to the wall. Remark that $h_{\text{opt}} = \left\lceil \frac{d(p,\mathcal{W})}{\mathcal{R}} \right\rceil$, where $d(p, \mathcal{W})$ is the (vertical) distance of p from the wall $\mathcal{W}$.

In the case where the protocol Π is randomized, or in the case where the distribution of the particles in the cloud is a random distribution, the number of hops h and the efficiency ratio C_h are random variables and one wishes to study their expected values.

The reason behind these definitions is that when p (or any intermediate particle in the information propagation to $\mathcal{W}$) "looks around" for a particle as near to $\mathcal{W}$ as possible to pass its information about $\mathcal{E}$, it may not get any particle in the perfect direction of the line vertical to $\mathcal{W}$. This difficulty comes mainly from three causes: (a) Due to the initial spreading of particles of the cloud in the area and because particles do not move, there might not be any particle in that direction. (b) Particles of sufficient remaining battery power may not be currently available in the right direction. (c) Particles may be there but temporarily "sleep" (i.e., not listen to transmissions) in order to save battery power.

Note that any given distribution of particles in the sensor cloud may not allow the ideal optimal number of hops to be achieved at all. In fact, the least possible number of hops depends on the input (the positions of the grain particles). We, however, compare the efficiency of protocols to the ideal case. A comparison with the best achievable number of hops in each input case will of course give better efficiency ratios for protocols.

To enable a first step toward a rigorous analysis of routing protocols, we make the following simplifying assumption: *The search phase always finds a* p'' (of sufficiently high battery) in the semicircle of center the particle p' currently possessing the information about the event and radius R, in the direction toward $\mathcal{W}$. Note that this assumption of always finding a particle can be relaxed in the following ways: (a) by repetitions of the search phase until a particle is found. This makes sense if at least one particle exists but was sleeping during the failed searches, (b) by considering, instead of just the semicircle, a cyclic sector defined by circles of radiuses

$\mathcal{R} - \Delta\mathcal{R}$, $\mathcal{R}$ and also take into account the density of the sensor cloud, and (c) if the protocol during a search phase ultimately fails to find a particle toward the wall, it may *backtrack*.

We also assume that the position of p'' is uniform in the arc of angle $2a$ around the direct line from p' vertical to $\mathcal{W}$. Each data transmission (one hop) takes constant time t (so the "hops" and time efficiency of our protocols coincide in this case). It is also assumed that each target selection is stochastically *independent* of the others, in the sense that it is always drawn uniformly randomly in the arc $(-a, a)$.

The above assumptions may not be very realistic in practice, however, they can be relaxed as explained above and in any case allow to perform a first effort toward providing some concrete analytical results.

Lemma 1 *The expected "hops efficiency" of the local target protocol in the a-uniform case is*

$$E(C_h) \simeq \frac{\alpha}{\sin\alpha}$$

for large h_{opt}. Also

$$1 \leq E(C_h) \leq \frac{\pi}{2} \simeq 1.57$$

for $0 \leq \alpha \leq \frac{\pi}{2}$.

Proof Due to the protocol, a sequence of points is generated, $p_0 = p, p_1, p_2, \ldots, p_{h-1}, p_h$ where p_{h-1} is a particle within $\mathcal{W}$'s range and p_h is beyond the wall. Let α_i be the (positive or negative) angle of p_i with respect to p_{i-1}'s vertical line to $\mathcal{W}$. It is

$$\sum_{i=1}^{h-1} d(p_{i-1}, p_i) \leq d(p, \mathcal{W}) \leq \sum_{i=1}^{h} d(p_{i-1}, p_i)$$

Since the (vertical) progress toward $\mathcal{W}$ is then $\Delta_i = d(p_{i-1}, p_i) = \mathcal{R}\cos\alpha_i$, we get

$$\sum_{i=1}^{h-1} \cos\alpha_i \leq h_{\text{opt}} \leq \sum_{i=1}^{h} \cos\alpha_i$$

From Wald's equation for the expectation of a sum of a random number of independent random variables (see [18]), then

$$E(h-1) \cdot E(\cos\alpha_i) \leq E(h_{\text{opt}}) = h_{\text{opt}} \leq E(h) \cdot E(\cos\alpha_i)$$

Now, $\forall i$, $E(\cos\alpha_i) = \int_{-\alpha}^{\alpha} \cos x \frac{1}{2\alpha} dx = \frac{\sin\alpha}{\alpha}$. Thus

$$\frac{\alpha}{\sin\alpha} \leq \frac{E(h)}{h_{\text{opt}}} = E(C_h) \leq \frac{\alpha}{\sin\alpha} + \frac{1}{h_{\text{opt}}}$$

Assuming large values for h_{opt} (i.e., events happening far away from the wall, which is the most non-trivial and interesting case in practice since the detection and propagation difficulty increases with distance) we get the result (since for $0 \leq \alpha \leq \frac{\pi}{2}$ it is $1 \leq \frac{\alpha}{\sin\alpha} \leq \frac{\pi}{2}$). □

12.2.3 Local Optimization: The Min-Two Uniform Targets Protocol (M2TP)

We can further assume that the search phase always returns *two points* p'', p''' each uniform in $(-\alpha, \alpha)$ and that a modified protocol M2TP selects the best of the two points, with respect to the local (vertical) progress. This is in fact an optimized version of the Local Target Protocol.

In a similar way as in the proof of the previous lemma, we can prove the following result:

Lemma 2 *The expected "hops efficiency" of the "min-two uniform targets" protocol in the a-uniform case is*

$$E(C_h) \simeq \frac{\alpha^2}{2(1-\cos\alpha)}$$

for $0 \leq \alpha \leq \frac{\pi}{2}$ *and for large h.*

Now remark that

$$\lim_{\alpha\to 0} E(C_h) = \lim_{\alpha\to 0} \frac{2\alpha}{2\sin a} = 1$$

and

$$\lim_{\alpha\to \frac{\pi}{2}} E(C_h) = \frac{(\pi/2)^2}{2(1-0)} = \frac{\pi^2}{8} \simeq 1.24$$

Thus, we have the following:

Lemma 3 *The expected "hops" efficiency of the min-two uniform targets protocol is*

$$1 \leq E(C_h) \leq \frac{\pi^2}{8} \simeq 1.24$$

for large h and for $0 \leq \alpha \leq \frac{\pi}{2}$.

Remark that, with respect to the expected hops efficiency of the local target protocol, the min-two uniform targets protocol achieves, because of the one additional search, a relative gain which is $(\pi/2 - \pi^2/8)/(\pi/2) \simeq 21.5\%$. Chatzigiannakis et al. [3] also experimentally investigate the further gain of additional (i.e., $m > 2$) searches.

12.2.4 Tight Upper Bounds to the Hops Distribution of the General Target Protocol

Consider the particle p (which senses the crucial event) at distance x from the wall. Let us assume that when p searches in the sector S defined by angles $(-\alpha, \alpha)$ and radius $\mathcal{R}$, another particle p' is returned *in the sector* with some probability density $f(\vec{p'})d\mathcal{A}$, where $\vec{p'} = (x_{p'}, y_{p'})$ is the position of p' in S and $d\mathcal{A}$ is an infinitesimal area around p'.

Definition 2 (Horizontal progress) Let Δx be the projection of the line segment (p, p') on the line from p vertical to $\mathcal{W}$.

We assume that each search phase returns such a particle, with independent and identical distribution $f()$.

Definition 3 (Probability of significant progress) Let $m > 0$ be the least integer such that $\mathbf{P}\left\{\Delta x > \frac{\mathcal{R}}{m}\right\} \geq p$, where $0 < p < 1$ is a given constant.

Lemma 4 For each continuous density $f()$ on the sector S and for any constant p, there is always an $m > 0$ as above.

Proof Remark that $f()$ defines a density function $\tilde{f}()$ on $(0, \mathcal{R}]$ which is also continuous. Let $\tilde{F}()$ its distribution function. Then we want $1 - \tilde{F}(\mathcal{R}/m) \geq p$, i.e., to find the first m such that $1 - p \geq \tilde{F}(\mathcal{R}/m)$. Such an m always exists since $\tilde{F}$ is continuous in $[0, 1]$. □

Definition 4 Consider the (discrete) stochastic process P in which with probability p the horizontal progress is $\mathcal{R}/m$ and with probability q it is zero, where $q = 1 - p$. Let Q the actual stochastic process of the horizontal progress implied by $f()$.

Lemma 5 $\mathbf{P}_P\{h \leq h_0\} \leq \mathbf{P}_Q\{h \leq h_0\}$

Proof The actual process Q makes always more progress than P. □

Now let $t = \left\lceil \frac{x}{\mathcal{R}/m} \right\rceil = \left\lceil \frac{mx}{\mathcal{R}} \right\rceil$. Consider the integer random variable H such that $\mathbf{P}\{H = i\} = q^i(1 - q)$ for any $i \geq 0$. Then H is geometrically distributed. Let $H_1, \ldots, H_t$ be t random variables, independent and identically distributed according to H. Clearly then

Lemma 6 $\mathbf{P}_{\mathrm{P}}\{\textit{number of hops is } \mathrm{h}\} = \mathbf{P}\{H_1 + \cdots H_{\mathrm{t}} = h\}$

The probability generating function of H is

$$H(s) = \mathbf{P}\{H = 0\} + \mathbf{P}\{H = 1\}s + \cdots + \mathbf{P}\{H = i\}s^i + \cdots$$

i.e.,

$$H(s) = p(1 + qs + q^2s^2 + \cdots + q^is^i + \cdots) \; = \; \frac{p}{1-qs}$$

But the probability generating function of $\sum_t = H_1 + \cdots + H_t$ is then just $\left(\frac{p}{1-qs}\right)^t$ by the convolution property of generating functions. This is just the generating function of the t-fold convolution of geometric random variables, and it is *exactly* the distribution of the *negative binomial distribution* (see [9], vol. I. p. 253). Thus,

Theorem 1 $\mathbf{P}_P\{\text{the number of hops is } h\} = \binom{-t}{h} p^t(-q)^h = \binom{t+h-1}{h} p^t q^h$

Corollary 1 *For the process P, the mean and variance of the number of hops are*

$$E(h) = \frac{tq}{p} \qquad \mathrm{Var}(h) = \frac{tq}{p^2}$$

Note that the method sketched above finds a distribution that upper bounds the number of hops till the crucial event is reported to the wall. Since for all $f()$ it is $h \geq \frac{x}{\mathcal{R}} = h_{\text{opt}}$ we get that

$$\frac{E_P(h)}{h_{\text{opt}}} \leq \frac{\left\lceil \frac{mx}{\mathcal{R}} \right\rceil \frac{q}{p}}{x/\mathcal{R}} \leq \frac{(m+1)q}{p}$$

Theorem 2 *The above upper bound process P estimates the expected number of hops to the wall with a guaranteed efficiency ratio* $\frac{(m+1)(1-p)}{p}$ *at most.*

Example 1 When for $p = 0.5$, we have $m = 2$ and the efficiency ratio is 3, i.e., the overestimate is three times the optimal number of hops.

12.3 PFR–A Probabilistic Multi-path Forwarding Protocol

The LTP protocol, as shown in the previous section, manages to be very efficient by always selecting exactly one next-hop particle, with respect to some optimization criterion. Thus, it tries to minimize the number of data transmissions. LTP is indeed very successful in the case of dense and robust networks, since in such networks a next-hop particle is very likely to be discovered. In sparse or faulty networks, however, the LTP protocol may behave poorly, because of many backtracks due to frequent failure to find a next-hop particle. To combine energy efficiency and fault tolerance, the Probabilistic Forwarding Protocol (PFR) has been introduced. The trade-offs in the performance of the two protocols implied above are shown and discussed in great detail in [5].

The PFR protocol was introduced in [4]. The authors assume the model M_1 (defined in Sect. 12.1.3).

12.3.1 The Protocol

The PFR protocol is inspired by the probabilistic multi-path design choice for the directed diffusion paradigm mentioned in [11]. The basic idea of the protocol (introduced in [4]) lies in probabilistically favoring transmissions toward the sink within a *thin zone* of particles around the line connecting the particle sensing the event $\mathcal{E}$ and the sink (see Fig. 12.4). Note that transmission along this line is energy optimal. However, it is not always possible to achieve this optimality, basically because certain sensors on this direct line might be inactive, either permanently (because their energy has been exhausted) or temporarily (because these sensors might enter a sleeping mode to save energy). Further reasons include (a) physical damage of sensors, (b) deliberate removal of some of them (possibly by an adversary in military applications), (c) changes in the position of the sensors due to a variety of reasons (weather conditions, human interaction etc.), and (d) physical obstacles blocking communication.

The protocol evolves in two phases.

Phase 1: The "Front" Creation Phase

Initially the protocol builds (by using a limited, in terms of rounds, flooding) a sufficiently large "front" of particles, in order to guarantee the survivability of the data propagation process. During this phase, each particle having received the data to be propagated, deterministically forward it toward the sink. In particular, and for a sufficiently large number of steps $s = 180\sqrt{2}$, each particle broadcasts the information to all its neighbors, toward the sink. Remark that to implement this phase, and in particular to count the number of steps, we use a counter in each message. This counter needs at most $\lceil \log 180\sqrt{2} \rceil$ bits.

Phase 2: The Probabilistic Forwarding Phase

During this phase, each particle P possessing the information under propagation calculates an angle ϕ by calling the subprotocol "ϕ-calculation" (see description below) and broadcasts $info(\mathcal{E})$ to all its neighbors with probability $\mathbf{P}_{\text{fwd}}$ (or it does not propagate any data with probability $1 - \mathbf{P}_{\text{fwd}}$) defined as follows:

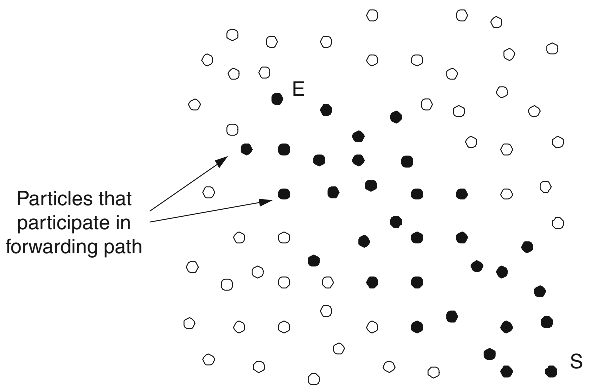


Fig. 12.4 Thin zone of particles

$$\mathbf{P}_{\text{fwd}} = \begin{cases} 1 & \text{if } \phi \geq \phi_{\text{threshold}} \\ \frac{\phi}{\pi} & \text{otherwise} \end{cases}$$

where ϕ is the angle defined by the line EP and the line PS and $\phi_{\text{threshold}} = 134°$ (the selection reasons of this $\phi_{\text{threshold}}$ can be found in [4]).

In both phases, if a particle has already broadcast $info(\mathcal{E})$ and receives it again, it ignores it. Also the PFR protocol is presented for a single event tracing. Thus no multiple paths arise and packet sizes do not increase with time.

Remark that when $\phi = \pi$ then P lies on the line ES and vice versa (and always transmits).

If the density of particles is appropriately high, then for a line ES there is (with high probability) a sequence of points "closely surrounding ES" whose angles ϕ are larger than $\phi_{\text{threshold}}$ and so that successive points are within transmission range. All such points broadcast and thus essentially they follow the line ES (see Fig. 12.4).

12.3.1.1 The ϕ-Calculation Subprotocol

Let P_{prev} the particle that transmitted $info(E)$ to P (see Fig. 12.5).

(1) When P_{prev} broadcasts $info(E)$, it also attaches the info $|EP_{\text{prev}}|$ and the direction $\overrightarrow{P_{\text{prev}}E}$.
(2) P estimates the direction and length of line segment $P_{\text{prev}}P$, as described in the model.
(3) P now computes angle $(\widehat{EP_{\text{prev}}P})$ and computes $|EP|$ and the direction of $\overrightarrow{PE}$ (this will be used in further transmissions from P).
(4) P also computes angle $(\widehat{P_{\text{prev}}PE})$ and by subtracting it from $(\widehat{P_{\text{prev}}PS})$ it finds ϕ.

Notice the following:

(i) The direction and distance from activated sensors to E is inductively propagated (i.e., P becomes P_{prev} in the next phase).
(ii) The protocol needs only messages of length bounded by $\log A$, where A is some measure of the size of the network area, since (because of (i) above) there is no cumulative effect on message lengths.

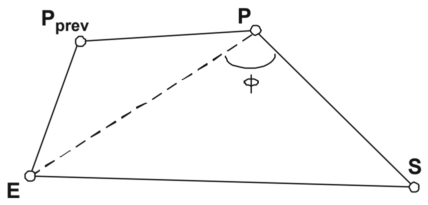


Fig. 12.5 Angle ϕ calculation example

Essentially, the protocol captures the intuitive, deterministic idea "if my distance from ES is small, then send, else do not send." Chatzigiannakis et al. [4] have chosen to enhance this idea by random decisions (above a threshold) to allow some local flooding to happen with small probability and thus to cope with local sensor failures.

12.3.2 Properties of PFR

Any protocol Π solving the data propagation problem must satisfy the following three properties:

- *Correctness*. Π must guarantee that data arrive to the position S, given that the whole network exists and is operational.
- *Robustness*. Π must guarantee that data arrive at enough points in a small interval around S, in cases where part of the network has become inoperative.
- *Efficiency*. If Π activates k particles during its operation then Π should have a small ratio of the number of activated over the total number of particles $r = \frac{k}{N}$. Thus r is an energy efficiency measure of Π.

We show that this is indeed the case for PFR.

Consider a partition of the network area into small squares of a fictitious grid G (see Fig. 12.6). Let the length of the side of each square be l. Let the number of squares be q. The area covered is bounded by ql^2. Assuming that we randomly throw in the area at least $\alpha q \log q = N$ particles (where $\alpha > 0$ a suitable constant), then the probability that a particular square is avoided tends to 0. So with very high probability (tending to 1) all squares get particles.

We condition all the analysis on this event, call it F, of at least one particle in each square.

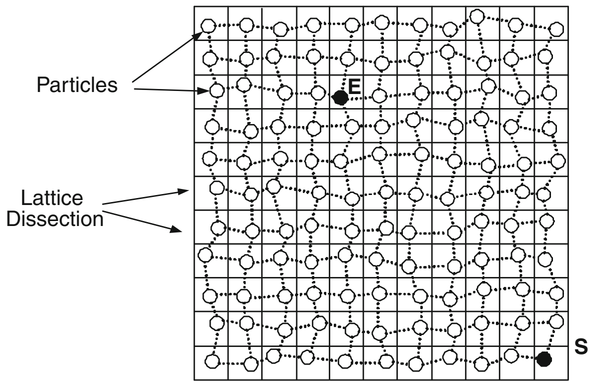


Fig. 12.6 A lattice dissection G

12.3.3 *The Correctness of* PFR

Without loss of generality, we assume each square of the fictitious lattice G to have side length 1.

Lemma 7 PFR *succeeds with probability 1* in sending the information from E to S given the event F.

Proof In the (trivial) case where $|ES| \leq 180\sqrt{2}$, the protocol is clearly correct due to front creation phase.

Let Σ a unit square of G intersecting ES in some way (see Fig. 12.7). Since a particle always exists somewhere in Σ, we will only need to examine the worst case of it being in one of the corners of Σ. Consider vertex A. The line EA is always to the left of AB (since E is at end of ES). The same is true for AS (S is to the right of B').

Let AD the segment from A perpendicular to ES (and D its intersection point) and let yy' be the line from A parallel to ES. Then,

$$\begin{aligned}
\phi = (\widehat{EAS}) &= 180^\circ - (\widehat{yAE}) - (\widehat{y'AS}) \\
&= 90^\circ - (\widehat{yAE}) + 90^\circ - (\widehat{y'AS}) \\
&= (\widehat{yAD}) - (\widehat{yAE}) + (\widehat{DAy'}) - (\widehat{y'AS})
\end{aligned}$$

Let $\widehat{\omega_1} = (\widehat{yAD}) - (\widehat{yAE})$ and $\widehat{\omega_2} = (\widehat{DAy'}) - (\widehat{y'AS})$ and, without loss of generality, let $ED < DS$. Then always $\widehat{\omega_1} > 45^\circ$, since it includes half of the 90°-angle of A in the unit square. Also

$$\sin(\widehat{y'AS}) = \sin(\widehat{ASD}) = \frac{AD}{AS} < \frac{AD}{DS}$$
$$\text{but} \quad AD \leq \sqrt{2} \quad (= AB) \quad \text{and} \quad DS \geq \frac{ES}{2} \geq 90\sqrt{2}$$
$$\Longrightarrow \sin(\widehat{y'AS}) \leq \frac{1}{90} \iff (\widehat{y'AS}) < 1^\circ$$

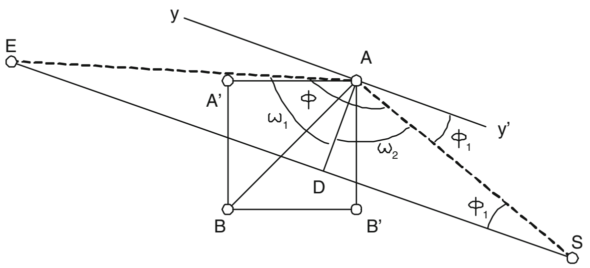


Fig. 12.7 The square Σ

(Note that here we use the elementary fact that when $x < 90°$ then $sin x < \frac{1}{90} \Longleftrightarrow x < 1°$). Then $\widehat{\omega_2} > 89°$, since $(\widehat{DAy'}) = 90°$ by construction. Thus

$$\phi = \widehat{\omega_1} + \widehat{\omega_2} > 45° + 89° = 134°$$

There are two other ways to place an intersecting to ES unit square Σ, whose analysis is similar.

But (a) the initial square (from E) always broadcasts due to the first protocol phase and (b) any intermediate intersecting square will be notified (by induction) and thus will broadcast. Hence, S is always notified, when the whole grid is operational. □

12.3.4 The Energy Efficiency of PFR

Consider the fictitious lattice G of the network area and let the event F hold. We have (at least) one particle inside each square. Now join all "nearby" particles (see Fig. 12.6) of each particle to it, thus by forming a new graph G' which is "lattice shaped" but its elementary "boxes" may not be orthogonal and may have varied length. When G's squares become smaller and smaller, then G' will look like G. Thus, for reasons of analytic tractability, we will assume in the sequel that our particles form a lattice (see Fig. 12.8). We also assume length $l = 1$ in each square, for normalization purposes. Notice, however, that when $l \to 0$ then "$G' \to G$" and thus all our results in this section hold for any random deployment "in the limit." We now prove the following:

Theorem 3 *The energy efficiency of the* PFR *protocol is* $\Theta\left(\left(\frac{n_0}{n}\right)^2\right)$ *where* $n_0 = |ES|$ *and* $n = \sqrt{N}$. *For* $n_0 = |ES| = o(n)$, *this is* $o(1)$.

Proof The analysis of the energy efficiency considers particles that are active but are as far as possible from ES. Thus the approximations we do are suitable for remote particles.

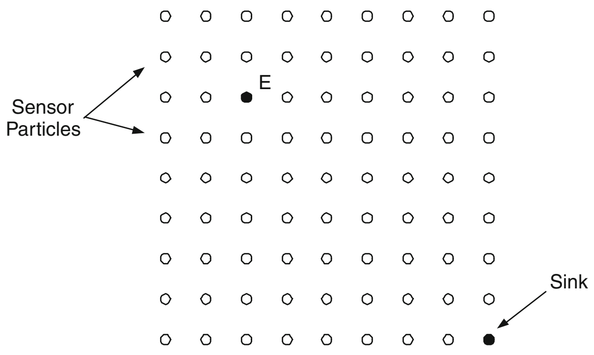


Fig. 12.8 A lattice sensor network

Here we estimate an upper bound on the number of particles in an $n \times n$ (i.e., $N = n \times n$) lattice. If k is this number then $r = \frac{k}{n^2}$ $(0 < r \leq 1)$ is the "energy efficiency ratio" of PFR.

We want r to be less than 1 and as small as possible (clearly, $r = 1$ leads to flooding). Since, by Lemma 7, ES is always "surrounded" by active particles, we will now assume without loss of generality that ES is part of a horizontal grid line (see Fig. 12.9) somewhere in the middle of the lattice.

Recall that $|ES| = n_0$ particles of all the active, via PFR, particles that continue to transmit, and let Q be a set of points whose shortest distance from particles in ES is maximum. Then L_Q is the locus (curve) of such points Q. The number of particles included in L_Q (i.e., the *area* inside L_Q) is the number k.

Now, if the distance of such points Q of L_Q from ES is ω then $k \leq 2(n_0 + 2\omega)\omega$ (see Fig. 12.10) and thus $r \leq \frac{2\omega(n_0+2\omega)}{n^2}$. Notice, however, that ω is a random variable hence we will estimate its expected value $\mathbf{E}(\omega)$ and the moment $\mathbf{E}(\omega^2)$ to get

$$\mathbf{E}(r) \leq \frac{4\mathbf{E}(\omega^2)}{n^2} + \frac{2n_0}{n^2}\mathbf{E}(\omega) \tag{12.1}$$

Now, look at points $Q : (\widehat{EQS}) = \phi < 30°$, i.e., $\phi < \frac{\pi}{6}$. We want to use the approximation $|ED| \simeq x$, where x is a random variable depending on the particle distribution.

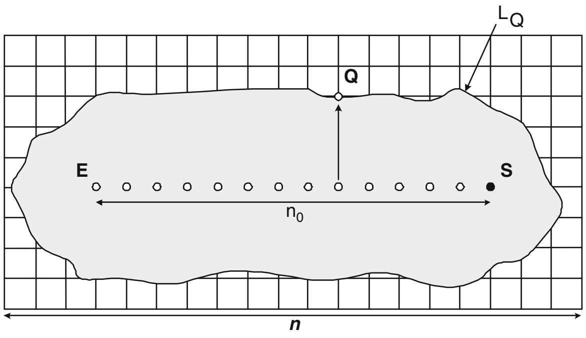


Fig. 12.9 The L_Q area

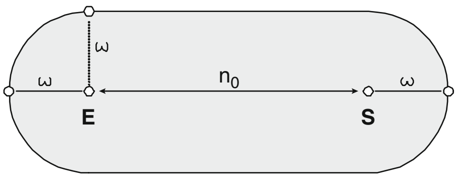


Fig. 12.10 The particles inside the L_Q area

Let $\frac{|ED|}{x} = 1 + \varepsilon$. Then a bound on the approximation factor ε determines a bound on the "cut-off" angle ϕ_0 since $|ED| \cos\left(\frac{\phi_o}{2}\right) = \varepsilon x$.

$$(1+\varepsilon)\, x \, \cos\left(\frac{\phi_o}{2}\right) = x \implies \cos\left(\frac{\phi_o}{2}\right) = \frac{1}{1+\varepsilon}$$

ϕ_0 above is constant that can be appropriately chosen by the protocol implementer. We chose $\phi_0 = 30°$ so $\varepsilon = 0.035$. Also, we remark here that the energy spent increases with $x_0 = n_0\left(1 - \frac{\xi}{2}\right)$ in the area below x_0, where $\xi = \tan 75°$, but decreases with x for $x > x_0$. This is a trade-off and one can carefully estimate the angle ϕ_0 (i.e., ϵ) to minimize the energy spent.
Let $\phi_1 = (\widehat{EQD})$, $\phi_2 = (\widehat{DQS})$ where $QD \perp ES$ and D in ES. In Fig. 12.11, $\phi = \phi_2 - \phi_1$.
We approximate ϕ by $\sin\phi_2 - \sin\phi_1$ (note: $\phi \simeq \sin\phi \geq -\sin\phi_1 + \sin\phi_2$).

$$\text{Note} \quad \sin\phi_1 \simeq \frac{|ED|}{|QE|} \simeq \frac{|ED|}{x} \quad \text{and} \quad \sin\phi_2 \simeq \frac{|DS|}{x}$$
$$\implies \sin\phi \simeq \sin\phi_2 - \sin\phi_1 \simeq \frac{|ES|}{x} = \frac{n_0}{x}$$

Thus $\frac{\phi}{\pi} \simeq \frac{n_0}{\pi x}$ for such points Q. We note that the analysis is very similar when $(\widehat{QES}) < \frac{\pi}{2}$ and thus $\phi = \phi_1 + \phi_2$.

Let $\mathcal{M}$ be the stochastic process that represents the vertical (to ES) distance of active points from ES, and $\mathcal{W}$ be the random walk on the vertical line yy' (see Fig. 12.12) such that, when the walk is at distance $x \geq x_0$ from ES then it (a) goes to $x + 1$ with probability $\frac{n_0}{\pi x}$ or (b) goes to $x - 1$ with probability $1 - \frac{n_0}{\pi x}$ and never goes below x_0, where x_0 is the "30°-distance" i.e. $x_0 = \frac{n_0 \xi}{2}$.

Clearly $\mathcal{W}$ dominates M, i.e., $\mathbf{P}_{\mathcal{M}}\{x \geq x_1\} \leq \mathbf{P}_{\mathcal{W}}\{x \geq x_1\}, \forall x_1 > x_0$.

Furthermore, $\mathcal{W}$ is dominated by the continuous time "discouraged arrivals" birth-death process $\mathcal{W}'$ (for $x \geq x_0$) where the rate of going from x to $x + 1$ in $\mathcal{W}'$ is $\frac{\alpha}{x} = \frac{n_0/\pi}{x}$ and the rate of returning to $x - 1$ is $1 - \frac{n_0}{\pi x_0} = 1 - \frac{2}{\pi\xi} = \mathbb{B}$.

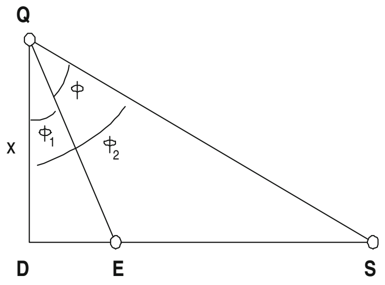


Fig. 12.11 The QES triangle

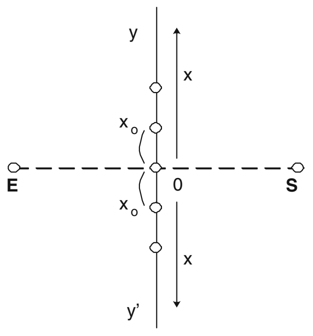


Fig. 12.12 The random walk $\mathcal{W}$

We know from [15] that for $\Delta x = x - x_0$ $(x > x_0)$

$$\mathbf{E}_{\mathcal{W}'}(\Delta x) = \frac{\alpha}{\mathbb{B}} = \frac{n_0}{\pi\left(1 - \frac{2}{\pi\xi}\right)}$$

Thus, $\mathbf{E}_{\mathcal{W}'}(x) = x_0 + \frac{\alpha}{\mathbb{B}}$, hence by domination

$$\mathbf{E}(\omega) \le \mathbf{E}_{\mathcal{M}}(x) \le x_0 + \frac{\alpha}{\mathbb{B}} \tag{12.2}$$

Also from [15] the process $\mathcal{W}'$ is a Poisson one and $\mathbf{P}\{\Delta x = k\} = \frac{(\alpha/\mathbb{B})^k}{k!} e^{-(\alpha/\mathbb{B})}$. From this, the variance of Δx is $\sigma^2 = \alpha/\mathbb{B}$ (again), i.e., for $\omega = x_0 + \Delta x$

$$\begin{aligned}
\mathbf{E}_{\mathcal{W}'}(\omega^2) &= \mathbf{E}_{\mathcal{W}'}\Big((x_0 + \Delta x)^2\Big)\\
&= x_0^2 + 2x_0\mathbf{E}_{\mathcal{W}'}(\Delta x) + \mathbf{E}_{\mathcal{W}'}(\Delta x^2)\\
&= x_0^2 + 2\frac{\alpha}{\mathbb{B}}x_0 + \Big(\sigma^2 + \mathbf{E}^2(\Delta x)\Big)\\
&= x_0^2 + 2\frac{\alpha}{\mathbb{B}}x_0 + \frac{\alpha}{\mathbb{B}} + \left(\frac{\alpha}{\mathbb{B}}\right)^2
\end{aligned}$$

$$\text{So} \quad \mathbf{E}_{\mathcal{W}'}(\omega^2) = \frac{3n_0^2}{4} + 2\,\frac{\alpha}{\mathbb{B}}\,\frac{n_0\,\xi}{2} + \frac{\alpha}{\mathbb{B}} + \left(\frac{\alpha}{\mathbb{B}}\right)^2$$

$$\text{where} \quad \frac{\alpha}{\mathbb{B}} = \frac{n_0}{\pi\left(1 - \frac{2}{\pi\,\xi}\right)} = \frac{n_0}{\tau}$$

$$\text{and} \quad \tau = \pi\left(1 - \frac{2}{\pi\,\xi}\right) = \pi - \frac{2}{\xi}$$

$$\text{thus} \quad \mathbf{E}_{\mathcal{W}'}(\omega^2) = n_0^2\left(\frac{3}{4} + \frac{\xi}{\tau} + \frac{1}{\tau^2}\right) + \frac{n_0}{\tau} \tag{12.3}$$

and by domination $\mathbf{E}(\omega^2) \le \mathbf{E}_{\mathcal{M}}(\omega^2) \le \mathbf{E}_{\mathcal{W}'}(\omega^2)$. So, finally

$$\mathbf{E}(r) \leq \frac{4\left[n_0^2\left(\frac{3}{4}+\frac{\xi}{\tau}+\frac{1}{\tau^2}\right)+\frac{n_0}{\tau}\right]}{n^2}+\frac{n_0^2}{n^2}\left(\xi+\frac{2}{\tau}\right) \tag{12.4}$$

which proves the theorem. □

12.3.5 *The Robustness of* PFR

We now consider the robustness of our protocol, in the sense that PFR must guarantee that data arrive at enough points in a small interval around S, in cases where part of the network has become inoperative, without, however, isolating the sink.

Lemma 8 PFR *manages to propagate the crucial data across lines parallel to ES, and of constant distance, with fixed nonzero probability (not depending on n, $|ES|$).*

Proof Here we consider particles very near the line ES. Thus the approximations that we do are suitable for nearby particles. Let Q' an active particle at vertical distance x from ES and $D \in ES$ such that $Q'D \perp ES$.

Let $yy' \parallel ES$, drawn from Q' (see Fig. 12.13) and $\phi_1 = (\widehat{yQ'E})$, $\phi_2 = (\widehat{SQ'y'})$. Then $\phi = 180° - (\phi_1 + \phi_2)$, i.e., $\frac{\phi}{\pi} = 1 - \frac{\phi_1+\phi_2}{\pi}$.

Since ϕ_1, ϕ_2 are small (for small x) we use the approximation $\phi_1 \simeq \sin\phi_1$ and $\phi_2 \simeq \sin\phi_2$

$$\phi_1 + \phi_2 \simeq \sin\phi_1 + \sin\phi_2 = \frac{x}{EQ'} + \frac{x}{Q'S}$$

Since $\phi > 90°$, ES is the biggest edge of triangle $(\widehat{EQ'S})$, thus $EQ' \leq n_0$ and $Q'S \leq n_0$. Hence,

$$\sin\phi_1 + \sin\phi_2 \leq \frac{2x}{n_0}$$

$$\text{and} \quad 1 - \frac{\sin\phi_1 + \sin\phi_2}{\pi} \geq 1 - \frac{2x}{\pi n_0} \quad (\text{for small } x)$$

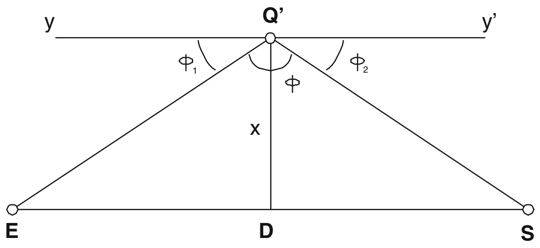


Fig. 12.13 The $Q'ES$ triangle

Without loss of generality, assume ES is part of a horizontal grid line. Here we study the case in which some of the particles on ES or very near ES (i.e. at angles $> 134^\circ$) are not operating.

Consider now a horizontal line ℓ in the grid, at distance x from ES. Clearly, some particles of ℓ near E will be activated during the initial broadcasting phase (see Phase 1).

Let $\mathcal{A}$ be the event that all the particles of ℓ (starting from p) will be activated, until a vertical line from S is reached. Then

$$\mathbf{P}(\mathcal{A}) \geq \left(1 - \frac{2x}{\pi n_0}\right)^{n_0} \simeq e^{-\frac{2x}{\pi}} \qquad \square$$

12.4 An Experimental Comparison of LTP, PFR

We evaluate the performance of four protocols (PFR, LTP, and two variations of LTP) by a comparative experimental study. The protocols have been implemented as C++ classes using the data types for two-dimensional geometry of LEDA [17] based on the environment developed in [3, 6]. Each class is installed in an environment that generates sensor fields given some parameters (such as the area size of the field, the distribution function used to drop the particles) and performs a network simulation for a given number of repetitions, a fixed number of particles, and certain protocol parameters.

In the experiments, we generate a variety of sensor fields in a 100 m $\times$ 100 m square. In these fields, we drop $n \in [100, 3000]$ particles randomly uniformly distributed on the sensor cloud plane, i.e., for density $0.01 \leq d \leq 0.3$. Each sensor particle has a fixed radio range of $\mathcal{R} = 5m$ and $\alpha = 90^\circ$. The particle p that initially senses the crucial event is always explicitly positioned at $(x, y) = (0, 0)$ and the sink is located at $(x, y) = (100, 100)$. Note that this experimental setup is based on and extends that used in [10, 13, 16]. We repeated each experiment for more than 5,000 times in order to achieve good average results.
We now define the efficiency measures investigated.

Definition 5 Let h_A (for "active") be the *number of "active" sensor particles* participating in the data propagation and let E_TR be the *total number of data transmissions* during propagation. Let T be the *total time* for the propagation process to reach its final position and H the *total number of "hops"* required.

Clearly, by minimizing h_A we succeed in avoiding flooding and thus we minimize energy consumption. Remark that in LTP we count as active those particles that transmit *info($\mathcal{E}$)* at least once.

Note that h_A, E_TR, T, and H are random variables. Furthermore, we define the success probability of the algorithm where we call success the eventual data propagation to the sink.

Definition 6 Let $\mathbf{P}_s$ be the *success probability* of the protocol.

We also focus on the study of the following parameter. Suppose that the data propagation fails to reach the sink. In this case, it is very important to know *"how close" to the sink* it managed to get. Propagation reaching close to the sink might be very useful, since the sink (which can be assumed to be mobile) could itself move (possibly by performing a random walk) to the final point of propagation and get the desired data from there. Even assuming a fixed sink, proximity to it is important, since the sink might in this case begin some "limited" flooding to get to where data propagation stopped. Clearly, the closer to the sink we get, the cheaper the flooding becomes.

Definition 7 Let F be the final position of the data propagation process. Let $\mathcal{D}$ be $\mathcal{F}$'s (Euclidean) distance from the sink $\mathcal{S}$.

Clearly in the case of total success $\mathcal{F}$ coincides with the sink and $\mathcal{D} = 0$.

We start by examining the success rate of the four protocols (see Fig. 12.14), for different particle densities. Initially, when the density is low (i.e., $d \leq 0.06$), the protocols fail to propagate the data to the sink. However, as the density increases, the success rate increases quite fast and for high densities all four protocols almost always succeed in propagating the data to the sink. Thus, all protocols are very successful. We remark a similar shape of the success rate function in terms of density. This is due to the fact that all protocols use local information to decide how to proceed by basically selecting (all protocols) the next particle with respect to a similar criterion (best progress toward the sink).

In the case when the protocols fail to propagate the data to the sink, we examine *"how close" to the sink* they managed to get. Fig. 12.15 depicts the distance of the final point of propagation to the position of the sink. Note that this figure should

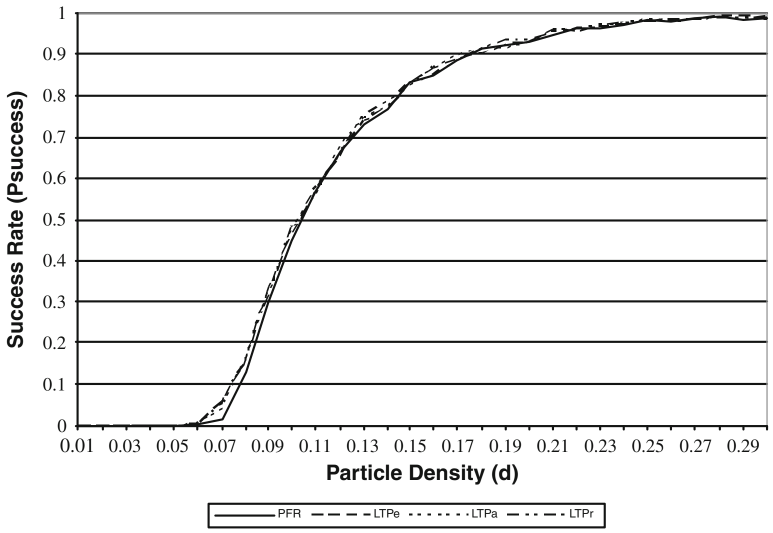


Fig. 12.14 Success probability ($\mathbf{P}_s$) over particle density $d = [0.01, 0.3]$

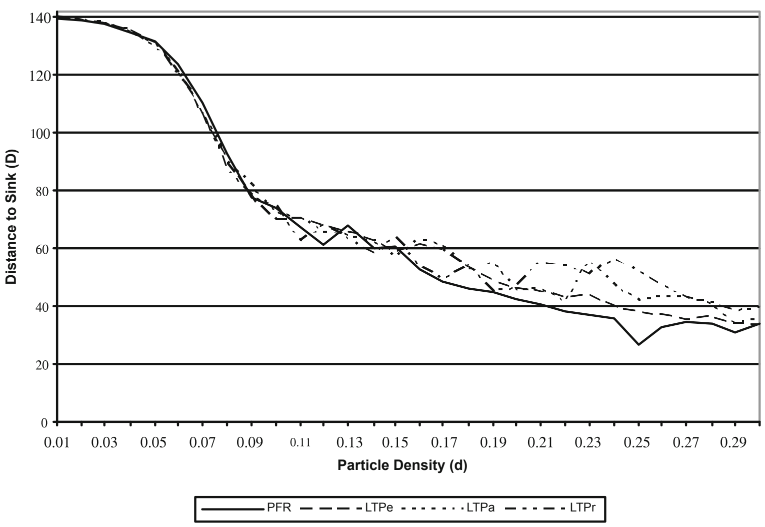


Fig. 12.15 Average distance from the sink ($\mathcal{D}$) over particle density $d = [0.01, 0.3]$

be considered in conjunction with Fig. 12.14 on the success rate. Indeed, failures to reach the sink are very rare and seem to appear in very extreme network topologies due to bad particle distribution in the network area.

Figure 12.16 depicts the ratio of active particles over the total number of particles ($r = \frac{h_A}{n}$) that make up the sensor network. In this figure we clearly see that PFR, for low densities (i.e., $d \leq 0.07$), indeed activates a small number of particles (i.e.,

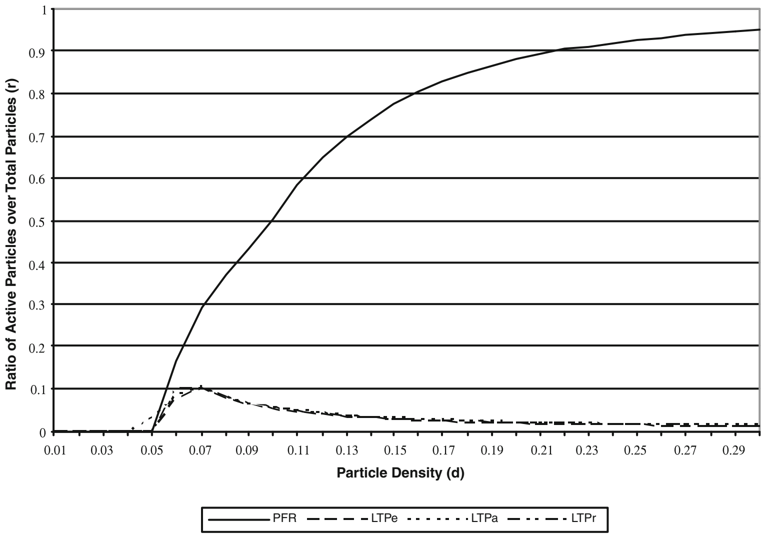


Fig. 12.16 Ratio of active particles over total particles (r) over particle density $d = [0.01, 0.3]$

$r \leq 0.3$) while the ratio r increases fast as the density of the particles increases. The LTP-based protocols seem more efficient and the ratio r seems to be independent of the total number of particles (because only one particle in each hop becomes active).

Remark 1 Because of the way PFR attempts to avoid flooding (by using angle ϕ to capture "distance" from optimality) its merits are not sufficiently shown in the setting considered here. We expect PFR to behave significantly better with respect to energy in much larger networks and in cases where the event is sensed in an average place of the network. Also, stronger probabilistic choices (i.e., $\mathbf{P}_{\text{fwd}} = \left(\frac{\phi}{\pi}\right)^{\alpha}$, where $a > 1$ a constant) may further limit the number of activated particles.

Furthermore, examining the total number of transmissions performed by the particles (see Fig. 12.17), it is evident that because the LTP-based protocols activate a small number of particles, the overall transmissions are kept low. This is a surprising finding, since the PFR protocol was originally designed to work without the need of any control messages so that the energy consumption is low. However, the comparative study clearly shows that avoiding the use of control messages does not achieve the expected results. So, even though all four protocols succeed in propagating the data, it is evident that the LTP-based protocols are more energy efficient in the sense that less particles are involved in the process.

We continue with the following two parameters: (a) the "hops" efficiency and (b) the time efficiency, measured in terms of rounds needed to reach the sink. As can be seen in Fig. 12.18, all protocols are very efficient, in the sense that the number of hops required to get to the sink tends below 40 even for densities $d \geq 0.17$. The value 40 in our setting is close to optimality since in an ideal placement, the diagonal line is of length $100\sqrt{2}$ and since for the transmission range $\mathcal{R} = 5$ the optimal number of hops (in an ideal case) is roughly 29. In particular PFR achieves

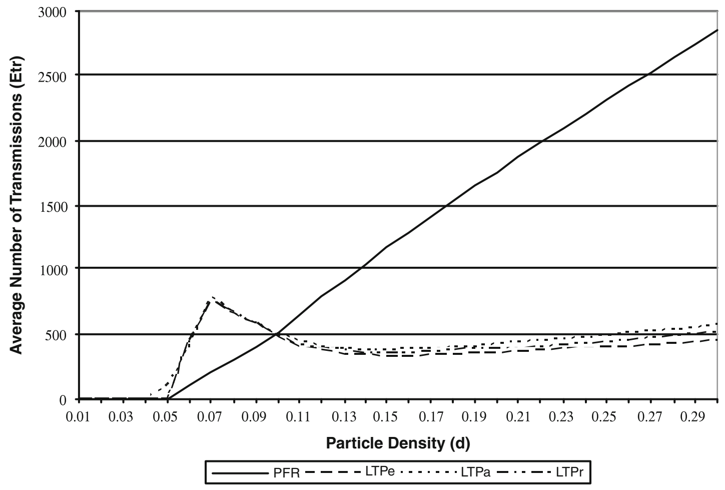


Fig. 12.17 Average number of transmissions (E_{TR}) over particle density $d = [0.01, 0.3]$

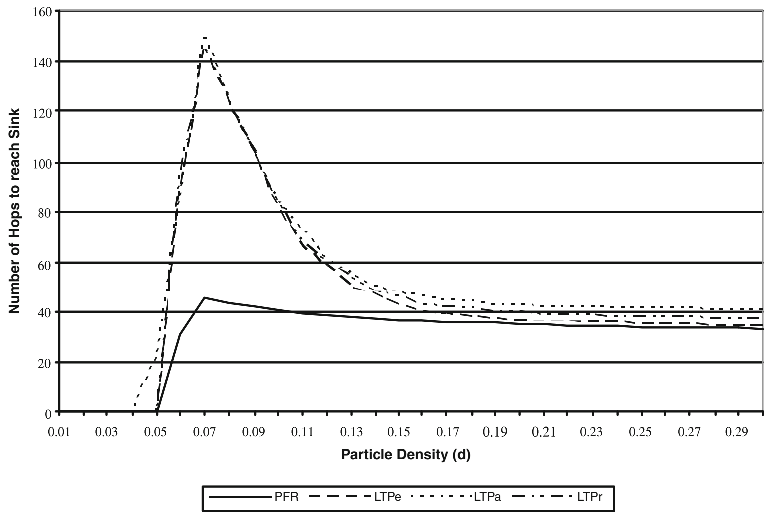


Fig. 12.18 Average number of hops to reach the sink (H) over particle density $d = [0.01, 0.3]$

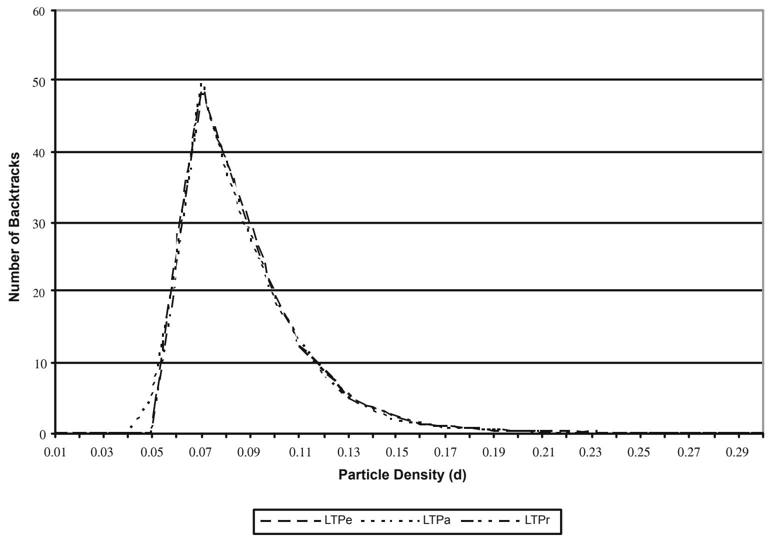


Fig. 12.19 Average number of backtracks over particle density $d = [0.01, 0.3]$

this for very low densities ($d \geq 0.07$). On the other hand, the LTP-based protocols exhibit a certain pathological behavior for low densities (i.e., $d \leq 0.12$) due to a high number of executions of the backtrack mechanism in the attempt to find a particle closer to the sink (see also Fig. 12.19).

Finally, in Fig. 12.19 we compare the three LTP-based protocols and the number of backtracks invoked in the data propagation. It is evident that for very low particle densities (i.e., $d \leq 0.12$), all three protocols perform a large number of backtracks in order to find a valid path toward the sink. As the particle density increases, the number of backtrack reduces fast enough and almost reaches zero.

12.5 Conclusions

We investigated some important aspects of online energy optimization in sensor networks, like minimizing the total energy spent in the network, minimizing the number (or the range) of data transmissions, combining energy efficiency and fault tolerance (by allowing redundant data transmissions which, however, should be optimized to not spend too much energy). Since it is very difficult (if possible at all) to achieve all the above goals at the same time we presented two characteristic protocols, each of which focuses on some of the energy efficiency goals above (while still performing well with respect to the rest goals as well). In particular, we presented: (a) The Local Target Protocol (LTP) that performs a local optimization trying to minimize the number of data transmissions and (b) the Probabilistic Forwarding Protocol (PFR) that creates redundant data transmissions that are probabilistically optimized, to trade off energy efficiency with fault tolerance.

Open issues for further research include considering the impact on protocols' performance of other important types of faults (malicious, byzantine). Also, it is worth investigating other performance aspects like congestion (possibly using game-theoretic methods).

Acknowledgments This work has been partially supported by the IST Programme of the EU under contract number IST-2005-15964 (**AEOLUS**).

References

1. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *In the Journal of Computer Networks*, 38: 393–422, 2002.
2. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 102–114, August 2002.
3. I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis. Smart dust protocols for local detection and propagation. In: *Proceedings 2nd ACM Workshop on Principles of Mobile Computing* – POMC'2002, pages 9–16. Also, in the *ACM Mobile Networks (MONET) Journal, Special Issue on Algorithmic Solutions for Wireless, Mobile, Adhoc and Sensor Networks*, MONET 10 (1): 133–149, 2005.
4. I. Chatzigiannakis, T. Dimitriou, S. Nikoletseas, and P. Spirakis. A Probabilistic algorithm for efficient and robust data propagation in smart dust networks. *In the Proceedings of the 5th European Wireless Conference on Mobile and Wireless Systems beyond 3G (EW 2004)*, pages 344–350, 2004. Also, in the *Ad-Hoc Networks Journal, Elsevier*, 4(5): 621–635, 2006.
5. I. Chatzigiannakis, T. Dimitriou, M. Mavronicolas, S. Nikoletseas, and P. Spirakis. A comparative study of protocols for efficient data propagation in smart dust networks. In *Proceedings International Conference on Parallel and Distributed Computing* – EUPOPAR 2003 (Distinguished Paper), pages 1003–1016. Also in the *Parallel Processing Letters (PPL) Journal*, 13(4): 615–627, 2003.

6. I. Chatzigiannakis and S. Nikoletseas. A Sleep-awake protocol for information propagation in smart dust networks. In: *Proceedings 3rd Workshop on Mobile and Ad-Hoc Networks (WMAN)*–IPDPS Workshops, IEEE Press, page 225, 2003. Also, in the *ACM/Baltzer Journal of Mobile Networks and Applications* (MONET), 9(4): 319–332, 2004.
7. D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In: *Proceedings 5th ACM/IEEE International Conference on Mobile Computing* – MOBICOM'1999, IEEE Computer Society, Seattle.
8. H. Euthimiou, S. Nikoletseas, and J. Rolim. Energy balanced data propagation in wireless sensor networks. In: *Proceedings 4th International Workshop on Algorithms for Wireless, Mobile, Ad-Hoc and Sensor Networks (WMAN '04), IPDPS* 2004, 2004. Also, in the *Wireless Networks (WINET) Journal*, 12(6): 691–707, 2006.
9. W. Feller. An introduction to probability theory and its applications, volume I, Wiley, 1968.
10. W. R. Heinzelman, A. Chandrakasan, H. Balakrishnan: Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In: *Proceedings 33rd Hawaii International Conference on System Sciences* – HICSS'2000, pages 876–882, IEEE Computer Society, Maui, HI, USA.
11. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A scalable and robust communication paradigm for sensor networks. In: *Proceedings 6th ACM/IEEE International Conference on Mobile Computing* – MOBICOM'2000, pages 56–67, IEEE Computer Society, Boston.
12. C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva: Directed diffusion for wireless sensor networking. Extended version of [11].
13. C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann: Impact of network density on data aggregation in wireless sensor networks. Technical Report 01–750, University of Southern California Computer Science Department, November 2001.
14. J.M. Kahn, R.H. Katz, and K.S.J. Pister: Next Century Challenges. Mobile networking for smart dust. In: *Proceedings 5th ACM/IEEE International Conference on Mobile Computing*, pages 271–278, September 1999.
15. L. Kleinrock. Queueing systems, Theory, Wiley Volume. I, page 100, 1975.
16. A. Manjeshwar and D.P. Agrawal. TEEN: A routing protocol for enhanced efficiency in wireless sensor networks. In: *Proceedings 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, satellite workshop of *16th Annual International Parallel & Distributed Processing Symposium* – IPDPS'02, IEEE Computer Society, Fort Lauderdale, Florida.
17. K. Mehlhorn and S. Näher. *LEDA: A platform for combinatorial and geometric computing*. Cambridge University Press, Cambridge, 1999.
18. S. M. Ross. *Stochastic processes, 2nd Edition*. Wiley, 1995.
19. P. Triantafilloy, N. Ntarmos, S. Nikoletseas, and P. Spirakis. Nanopeer networks and P2P Worlds. In: *Proceedings 3rd IEEE International Conference on Peer-to-Peer Computing*, pages 40–46, Linkoeping, Sweden, 2003.

Chapter 13
Oblivious Routing for Sensor Network Topologies

Costas Busch, Malik Magdon-Ismail, and Jing Xi

Abstract We present oblivious routing algorithms whose routing paths are constructed independent of each other, with no dependence on the routing history. Oblivious algorithms are inherently adaptive to dynamic packet traffic, exhibit low congestion, and require low maintenance. All these attributes make oblivious algorithms to be suitable for sensor networks which are characterized by their limited energy and computational resources. Specifically, low congestion provides load balancing, and low stretch provides low-energy utilization. We present two simple oblivious routing algorithms. The first algorithm is for *geometric networks* in which nodes are embedded in the Euclidean plane. In this algorithm, a packet path is constructed by first choosing a random intermediate node in the space between the source and destination and then the packet is sent to its destination through the intermediate node. In the second algorithm we study *mesh networks*, where the nodes are arranged in a two-dimensional grid. Grids are interesting symmetric topologies which can be used as a testbed for designing efficient new routing algorithms in sensor networks. The oblivious algorithm in the mesh constructs the paths by decomposing the network into smaller submeshes in a hierarchical manner. This algorithm can be extended to d dimensions, which makes it suitable for three-dimensional sensor network deployments, such as in buildings and tall structures. We analyze the algorithms in terms of the stretch and congestion of the resulting paths and demonstrate that they exhibit near optimal performance.

13.1 Introduction

Routing algorithms specify the paths to be followed by packets in a network. A routing algorithm is *oblivious* if the path of every packet is given independently of the paths of the other packets and without considering the history of the previously routed packets. Oblivious algorithms are by their nature distributed and capable of solving online routing problems, where packets continuously arrive in the network.

C. Busch (✉)
Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803, USA
e-mail: busch@csc.lsu.edu

Hence, oblivious routing is often preferred to non-oblivious routing, since one does not need to make assumptions regarding the nature of the traffic.

In wireless and sensor network applications the nodes may have power and computing capability constraints. For example, a sensor node is typically operated with a battery that has limited energy capacity. To maximize the lifetime of the nodes, the time until nodes run out of power, it is important to minimize the utilization of individual nodes. This relates directly to balancing the packet traffic for minimizing the node congestion. With load balancing, the lifetime of a battery operated sensor network is prolonged, since the time that the first node runs out of energy is extended. Further, using paths of small stretch (ratio of path length to shortest path) is also beneficial to the sensor network since small stretch implies low overall energy utilization.

Oblivious routing algorithms are suitable for balancing the congestion in the network and therefore extending the lifetime of the nodes. They can also provide small stretch in interesting sensor network deployment scenarios. Oblivious algorithms are easy to implement in wireless and sensor networks, on account of their simplicity. We present two oblivious routing algorithms which are suitable for wireless sensor networks. The first algorithm is for *geometric networks*, while the second is for *mesh networks*. Both of these algorithms achieve low congestion and stretch. We continue with describing each of them.

13.1.1 Geometric Networks

We present the oblivious routing algorithm for geometric networks which was originally proposed in [7]. In geometric networks, the nodes are placed in the two-dimensional Euclidian space and we assume that all the nodes are contained in some geographic area $\mathcal{A}$ (Fig. 13.1). Suppose that a packet wants to go from a node s to a node t in the network. The algorithm is to choose a random intermediate node w in the space between the s and t, then sends the packet to w, and then sends the packet from w to its destination (see Fig. 13.3). In order to implement this idea, we assume that between every pair of nodes there is a dedicated path which we call the *default path*. For example, the default path between two nodes u and v could be a shortest path that connects them. We denote the set of all default paths by Q. The choice of the default paths affects the performance of our algorithm, and the closer the default paths are to the geodesics, the lines that connect the respective end points of the paths, the better the performance of the algorithm.

We analyze the algorithm in terms of stretch and congestion. Consider some set of paths P produced by the routing algorithm. Denote by $\mathsf{stretch}(P)$ the maximum ratio of a path length to the length of the respective shortest path (the length is measured in the number of node hops). The *node congestion* C_{node} is the maximum number of paths that use any node in the network. The *edge congestion* C_{edge} is the maximum number of paths that use any edge in the network. Let C^*_{node} and C^*_{edge} denote the optimal node and edge congestions, which could be obtained by a brute force search through all possible paths from the sources to the destinations in P.

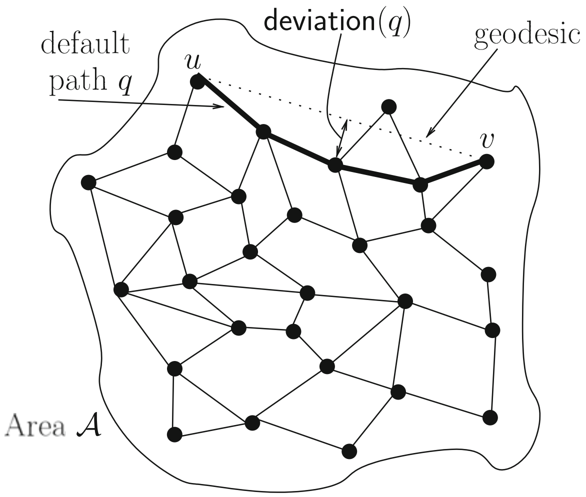


Fig. 13.1 Example of a geometric network

The stretch and congestion of the paths P produced by our algorithm depend on the quality of the default paths Q. In particular, provided that the geometric embedding is "faithful" to the topology of the network (i.e., nodes far apart are connected with more hops than nodes closer to each other) we obtain:

$$\mathsf{stretch}(P) = O\left(\mathsf{stretch}(Q)\right),$$
$$C_{\text{node}} = O\left(C^*_{\text{node}} \cdot (1 + \mathsf{deviation}^3(Q)) \cdot \log(n + \mathsf{deviation}(Q))\right)$$

where n is the number of nodes, and $\mathsf{deviation}(Q)$ measures the extent of deviation of the default paths from geodesics (see Fig. 13.1). We also obtain a corresponding result for the edge congestion. The congestion results hold with high probability, while the stretch result is deterministic.

We apply our general result to two particular geometric networks: the two-dimensional mesh network, and uniformly distributed disk networks. Both of these kinds of network have geometric embeddings that are faithful to the network topologies. The mesh network is a two-dimensional grid of nodes (see Fig. 13.2). In disk networks, each node is connected to any node within a specific disk radius. In the uniformly distributed disk graphs, each unit square area contains a constant number of nodes. In these networks, we can choose default paths with constant stretch and deviation. Therefore, our algorithm gives paths with *constant* stretch. We obtain node and edge congestions which are within logarithmic factors of optimal, $C_{\text{node}} = O\left(C^*_{\text{node}} \cdot \log n\right)$ and $C_{\text{edge}} = O\left(C^*_{\text{edge}} \cdot \log n\right)$, with high probability. Maggs et al. [21] give a worst case edge congestion lower bound of $\Omega\left(C^*_{\text{edge}} \cdot \log n\right)$ for any oblivious routing algorithm in the two-dimensional mesh. Therefore, in addition to

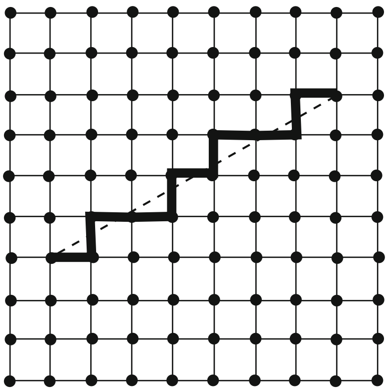


Fig. 13.2 The two-dimensional mesh network and a shortest path between two nodes

constant stretch, the congestion we obtain is optimal, within constant factors, for oblivious algorithms.

13.1.2 Mesh Networks

We continue with an alternative oblivious algorithm for mesh networks. A two-dimensional mesh with n nodes is simply a $\sqrt{n} \times \sqrt{n}$ grid of nodes (see Fig. 13.2). The oblivious algorithm for geometric networks that we described above can be applied to the two-dimensional mesh. However, it cannot be extended to higher dimensions. Here, we present the oblivious routing algorithm in the mesh which originally appears in [8]. The algorithm can be used for any d-dimensional mesh network ($d \geq 2$) and it provides near optimal congestion while maintaining a small stretch.

The benefit of a higher dimensional algorithm is that it can be used in sensor network deployments in buildings and other tall structures which are not restricted in the two-dimensional space. Note that the mesh topology has a symmetric topology and it may not be directly applicable in real deployment scenarios. However, the mesh topology exhibits many characteristics which can be found in real sensor networks (and especially in random uniform area deployment), such as the low node degree, small ratio of Euclidian to graph distance, and low doubling dimension. Routing algorithms on the mesh often generalize to other network topologies as well and typically the mesh is used as an exploration testbed for the design and analysis of new efficient algorithms.

Given a routing problem (collection of sources and destinations), let $C^* = C^*_{\text{edge}}$ denote the optimal edge congestion attainable by any routing algorithm (oblivious or not). We give an oblivious routing algorithm for the d-dimensional mesh with n nodes, that achieves congestion $O(d \cdot C^* \cdot \log n)$, and stretch $O(d^2)$, For the d-dimensional mesh with n nodes, Maggs et al. [21] give the lower bound $C^*_{\text{obl}} =$

$\Omega\left(\frac{C^*}{d} \cdot \log n\right)$ in the worst case for any oblivious routing algorithm. Considering the class of oblivious algorithms, our algorithm is within $O(d^2)$ of optimal for both congestion and dilation. For a fixed d, our algorithm is optimal to within constant factors.

Our algorithm is based upon a hierarchical decomposition of the mesh. Starting at its source node, a packet constructs its path by randomly selecting intermediate points in submeshes of increasing size until the current submesh contains the destination node. Random intermediate points are then selected in submeshes of decreasing size until the destination node is reached. The key new idea we introduce is the notion of "bridge" submeshes that make it possible to move from a source to a destination more quickly, without increasing the congestion. These bridge submeshes are instrumental in controlling the stretch, while maintaining low congestion.

13.2 Geometric Networks

13.2.1 Preliminaries on Geometric Networks

Consider a geometric network G with n nodes which is embedded in the Euclidean plane, $\mathcal{R}^2$ (see Fig. 13.1). We assume that G is un-weighted, undirected, connected, and stationary. Further, its edges are un-weighted, i.e., the communication cost of every link is 1 regardless of the link's Euclidian distance. Every node v_i has a position $\mathbf{x}_i \in \mathcal{R}^2$. We will also use the notation $\mathbf{x}(v)$ to denote the position of the node v. The network is defined over some *area* $\mathcal{A}$. We will also refer to the network itself as $\mathcal{A}$ when the context is clear. Thus, $\mathbf{x}_i \in \mathcal{A}$ for all i. For the area $\mathcal{A}$, we define a *coverage radius* $R(\mathcal{A})$ of the area as follows (we drop the $\mathcal{A}$ dependence when the context is clear). If, for every point $\mathbf{x} \in \mathcal{A}$, there is at least one node v that is located at most a (Euclidean) distance R from $\mathbf{x}$, then R is a coverage radius, i.e., from any point in $\mathcal{A}$, one needs to go a distance of at most R to reach some node in the network.

We define the *pseudo-convexity* $\gamma(\mathcal{A})$ of area $\mathcal{A}$ as follows. Let $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{A}$, and consider the line ℓ joining $\mathbf{x}_1$ to $\mathbf{x}_2$. Let $\ell^\perp$ be a line of equal length to ℓ such that ℓ and $\ell^\perp$ are mutually perpendicular bisectors. Let $\ell^\perp_\mathcal{A} \subseteq \ell^\perp$ be the intersection of $\ell^\perp$ with $\mathcal{A}$. Denote by $\left|\ell^\perp_\mathcal{A}\right|$ the measure or "length" of $\ell^\perp_\mathcal{A}$. We define the local pseudo-convexity at $\mathbf{x}_1, \mathbf{x}_2$ as $\gamma(\mathbf{x}_1, \mathbf{x}_2) = \left|\ell^\perp_\mathcal{A}\right| / |\ell^\perp|$. The pseudo-convexity γ of $\mathcal{A}$ is the infimum over all pairs $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{A}$ of $\gamma(\mathbf{x}_1, \mathbf{x}_1)$

$$\gamma = \infty_{\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{A}} \gamma(\mathbf{x}_1, \mathbf{x}_2)$$

In words, γ is a lower bound on the fraction of the perpendicular bisector $\ell^\perp$ that is guaranteed to be in $\mathcal{A}$. Note that $\mathcal{A}$ is convex if $\gamma \geq \frac{1}{2}$ but that the converse is not true (consider a very thin rectangle). For any regular convex polygon, or a circle, $\gamma \geq \frac{1}{2}$. For a network embedded in a fixed area $\mathcal{A}$, γ is independent of n, which will have important consequences on the optimality of our path selection algorithm (provided that $\gamma > 0$).

Since the network is embedded in $\mathcal{R}^2$, there are two notions of distance between two nodes u, v that are useful. The first is the *Euclidean distance*, $\mathsf{dist}_E(u, v)$ which is the length of the straight line (or *geodesic*) joining the positions $\mathbf{x}(u)$ and $\mathbf{x}(v)$. For two points $\mathbf{x}, \mathbf{y} \in \mathcal{R}^2$, $\| \mathbf{x} - \mathbf{y} \|$ is the Euclidean distance between them. Thus, $\mathsf{dist}_E(u, v) = \| \mathbf{x}(u) - \mathbf{x}(v) \|$. The second useful distance measure is the graph-theoretic or *network distance* $\mathsf{dist}_G(u, v)$ which is the length of the shortest path in G from u to v. For any path p in G, we use $|p|$ to denote the length of the path (number of edges in the path), and we define the *Euclidean path length* $|p|_E$ to be the weighted path length, where the weights on the edges are set to the Euclidean distance between the nodes they connect.

For two nodes u, v, we use the measure $\mathsf{dist}_G(u, v)/\mathsf{dist}_E(u, v)$ to represent how well the Euclidean distances in the network embedding represent the network distances. We introduce two parameters α, β to denote lower and upper bounds for this measure. Thus, for every pair of nodes u, v,

$$\alpha \le \frac{\mathsf{dist}_G(u, v)}{\mathsf{dist}_E(u, v)} \le \beta$$

Thus, two nodes u, v that are connected by an edge ($\mathsf{dist}_G(u, v) = 1$) cannot be separated by more than a distance of $\frac{1}{\alpha}$. Note also that $\mathsf{dist}_G(u, v) \ge 1$, so $\mathsf{dist}_E(u, v) \ge \frac{1}{\beta}$. We thus have the following useful lemma.

Lemma 1 *For any two nodes, u, v, $\mathsf{dist}_E(u, v) \ge \frac{1}{\beta}$. If u and v are adjacent then $\mathsf{dist}_E(u, v) \le \frac{1}{\alpha}$.*

Lemma 1 allows us to derive an upper bound on the number of nodes that can be in a disk.

Lemma 2 *Consider a disk of radius $r \ge \frac{1}{\beta}$ containing M nodes. Then $M \le c(\beta r)^2$, where c is a constant, $c \le 1 + \pi / \left(\frac{2\pi}{3} - \frac{\sqrt{3}}{2}\right)$.*

Proof The intuition is that every node accounts for an area of at least π/β^2. Since the total area is πr^2, there can be at most $\pi r^2/(\pi/\beta^2) = (\beta r)^2$ nodes. The only complication is that nodes near the boundary do not take up the entire area π/β^2, as part of this area could be outside the disk. Taking this boundary phenomenon into account gives us the constant c.

To prove the lemma, consider the circle of radius $r - \frac{1}{\beta}$ with M_1 nodes and the remaining ring from $r - \frac{1}{\beta}$ to r with M_2 nodes. Since every one of the M_1 nodes defines an area of radius $\frac{1}{\beta}$ that is completely enclosed in the disk, we have $M_1 \le (\beta r)^2$. Now consider the ring. The smallest area blocked off by a node occurs when the node is on the boundary, in which case the area is smallest when $r = \frac{1}{\beta}$. Some geometric considerations show that this area blocked off is at least $\frac{1}{\beta^2}\left(\frac{2\pi}{3} - \frac{\sqrt{3}}{2}\right)$, and since the area of the ring is at most πr^2, $M_2 \le (\beta r)^2 \pi / \left(\frac{2\pi}{3} - \frac{\sqrt{3}}{2}\right)$. To conclude, note that $M \le M_1 + M_2$. ⊠

For every pair of nodes u, v, we assume that a *default path* $q(u, v)$ in G is provided. For example, the default paths could be the shortest paths connecting the pairs of nodes. The default paths should actually have certain good properties and may not be shortest paths. Denote the set of all $n(n-1)$ default paths by the set Q. For a given default path $q(u, v)$, we define the stretch of the path, $\mathsf{stretch}(q)$, to be $|q(u, v)|/\mathsf{dist}_G(u, v)$ which is the factor by which q is longer than the shortest path between u and v.

Consider the infinite line ℓ drawn through the points $\mathbf{x}(u)$ and $\mathbf{x}(v)$. Let z be any intermediate node in the path $q(u, v)$. The *displacement* of z from ℓ is the perpendicular (Euclidean) distance from $\mathbf{x}(z)$ to ℓ. The deviation of $q(u, v)$ from ℓ, denoted $\mathsf{deviation}(q)$, is the maximum displacement of any intermediate node z of q from ℓ. In other words, $\mathsf{deviation}(q)$ measures how closely the path $q(u, v)$ stays to the straight line (geodesic) from $\mathbf{x}(u)$ to $\mathbf{x}(v)$.

The stretch factor for the entire set of paths Q is the maximum stretch of any path in Q and similarly with the deviation of Q. We use $\varSigma$ to denote the stretch and $\varDelta$ to denote the deviation:

$$\varSigma_Q = \mathsf{stretch}(Q) = \max_{q \in Q} \mathsf{stretch}(q)$$
$$\varDelta_Q = \mathsf{deviation}(Q) = \max_{q \in Q} \mathsf{deviation}(q)$$

As we will see later in the analysis of our path selection algorithm, if the default paths have small stretch and deviation, then the path selection performance is closer to optimal. Thus, it is beneficial to select default paths that make these parameters as small as possible. We will see later that for a variety of networks they can be made constants.

13.2.2 Oblivious Routing on Geometric Networks

Here we describe our oblivious routing algorithm. The task of the algorithm is to provide a path for each packet in the network. It is assumed that each node knows the default paths that connect it to other nodes in the network. The algorithm is randomized and we assume that each node has access to a sequence of random numbers. The path selection algorithm is executed for each packet independently of every other packet, so the algorithm is oblivious, and thus distributed and online. Algorithm 1 is the detailed algorithm for a particular packet. The algorithm is similar for any other packet. Figure 13.3 graphically illustrates the algorithm.

In the analysis of the algorithm, we consider a set of N packets $\varPi$ which we will refer to with their sources and destinations, $\varPi = \{s_i, t_i\}_{i=1}^{N}$. The result of applying the algorithm to each packet is a set of paths $P = \{p_i\}_{i=1}^{N}$, where each path $p_i \in P$ is from the source node s_i to the destination node t_i. We define the stretch for a path

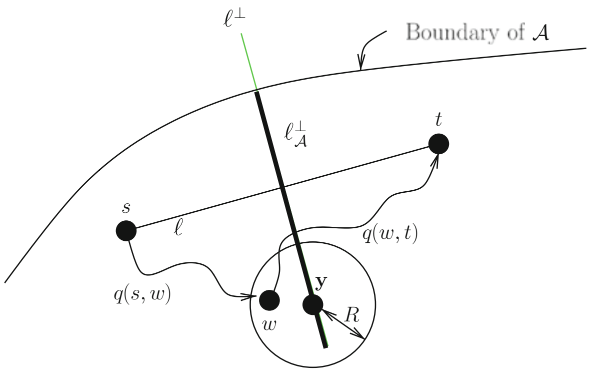


Fig. 13.3 Path selection with oblivious routing algorithm for geometric networks

Algorithm 1 Oblivious Routing for Geometric Networks

```
Input: A graph $G$ embedded in an area $\mathcal{A}$ with default paths $Q$; and a packet $\pi$ with source $s$ and destination $t$;
Output: A path $p(s,t)$ from $s$ to $t$;
1: Let $\ell$ be the geodesic line segment that connects $\mathbf{x}(s)$ and $\mathbf{x}(t)$. Let $\ell^\perp$ be the perpendicular bisector of $\ell$ which has the same length as $\ell$ and is also bisected by $\ell$. Let $\ell^\perp_{\mathcal{A}}$ be the part of $\ell^\perp$ inside $\mathcal{A}$;
2: Choose a point $\mathbf{y}$ randomly and uniformly on $\ell^\perp_{\mathcal{A}}$;
3: Find a node $w$ close to $\mathbf{y}$ within coverage radius $R$;
4: The path $p(s,t)$ from $s$ to $t$ is formed by concatenating the default paths $q(s,w)$ and $q(w,t)$:
       $p(s,t) = q(s,w)q(w,t);$
```

$p \in P$ as well as the stretch factor for the entire set P as we did in Sect. 13.2.1 with the default paths Q. We define D^* as the maximum shortest path length between any pair of sources and destinations in Π, namely, $D^* = \max_i \mathsf{dist}_G(s_i, t_i)$. We first analyze the stretch of paths P and then we continue with the node congestion and edge congestion.

13.2.2.1 Stretch Analysis

We now give a bound on $\mathsf{stretch}(P)$, the stretch factor of the paths selected.

Theorem 1 $\mathsf{stretch}(P) \le \frac{\sqrt{2}\beta}{\alpha} \cdot \Sigma_Q \cdot \left(1 + \sqrt{2}R\alpha\right)$.

Proof We will refer to Fig. 13.3 in our proof. By construction, $\sqrt{2}\| \mathbf{x}(s) - \mathbf{y} \| \le \| \mathbf{x}(s) - \mathbf{x}(t) \|$, and $\| \mathbf{x}(s) - \mathbf{y} \| = \| \mathbf{x}(t) - \mathbf{y} \|$. Since $\| \mathbf{x}(w) - \mathbf{y} \| \le R$, by the triangle inequality, we have that

$$\begin{aligned} \| \mathbf{x}(s) - \mathbf{x}(w) \| &\le \| \mathbf{x}(s) - \mathbf{y} \| + \| \mathbf{x}(w) - \mathbf{y} \| \\ &\le \tfrac{1}{\sqrt{2}} \| \mathbf{x}(s) - \mathbf{x}(t) \| + R \end{aligned}$$

Similarly,

$$\| \mathbf{x}(t) - \mathbf{x}(w) \| \leq \frac{1}{\sqrt{2}} \| \mathbf{x}(s) - \mathbf{x}(t) \| + R$$

From the definition of Σ_Q, the stretch factor of the default paths, we have

$$\begin{aligned} |q(s, w)| &\leq \Sigma_Q \cdot \mathsf{dist}_G(s, w) \\ &\leq \beta \cdot \Sigma_Q \cdot \mathsf{dist}_E(s, w) \end{aligned}$$

and similarly

$$|q(w, t)| \leq \beta \cdot \Sigma_Q \cdot \mathsf{dist}_E(w, t)$$

We thus conclude that

$$\begin{aligned} |p(s, t)| &= |q(s, w)| + |q(w, t)| \\ &\leq \beta \cdot \Sigma_Q \cdot (\| \mathbf{x}(s) - \mathbf{x}(w) \| + \| \mathbf{x}(t) - \mathbf{x}(w) \|) \\ &\leq \beta \cdot \Sigma_Q \cdot \left(\sqrt{2} \| \mathbf{x}(s) - \mathbf{x}(t) \| + 2R \right) \end{aligned}$$

Since $\| \mathbf{x}(s) - \mathbf{x}(t) \| \leq \frac{1}{\alpha} \mathsf{dist}_G(s, t)$, and $\mathsf{dist}_G(s, t) \geq 1$, we obtain the theorem.

Typically R, α, β are constants, in which case $\mathsf{stretch}(P) = O(\mathsf{stretch}(Q))$, i.e., the stretch factor of the algorithm is determined by the quality of the default paths.

13.2.2.2 Node Congestion Analysis

We now turn to the node congestion. We will get a bound on the expected congestion for any particular node with respect to the optimal congestion. We will then use a Chernoff bounding argument to obtain a high probability result.

To bound the expected node congestion for a particular node v, we need to understand the probability that a particular packet might use the node. Thus consider a particular packet π, with source s and destination t, which uses intermediate node w. Phase I of the path $p(s, t)$ corresponds to the first part $q(s, w)$, while phase II to the second part $q(w, t)$. Suppose that the packet uses v in phase I of its path (we will bound the probability that π uses v in phase I of its path, a similar argument applies to phase II of the path). Let r denote $\| \mathbf{x}(v) - \mathbf{x}(s) \|$. The situation is illustrated in Fig. 13.4. A circle of radius Δ_Q is drawn around v. We give an upper bound on the probability that π uses node v in the following lemma.

Lemma 3 *Suppose that packet π has source s and destination t. Let P_I be the probability that π uses node v in phase I of its path and P_{II} be the probability that π uses node v in phase II of its path. Then,*

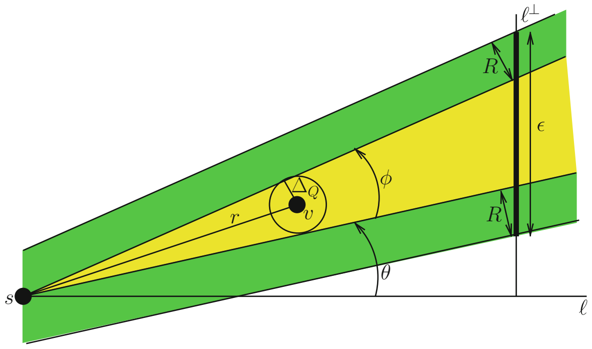


Fig. 13.4 Probability of using a node v

$$P_I \le \frac{5}{\gamma}\left(\frac{R}{\|\mathbf{x}(s)-\mathbf{x}(t)\|}+\frac{\Delta_Q}{\|\mathbf{x}(s)-\mathbf{x}(v)\|}\right)$$
$$P_{II} \le \frac{5}{\gamma}\left(\frac{R}{\|\mathbf{x}(s)-\mathbf{x}(t)\|}+\frac{\Delta_Q}{\|\mathbf{x}(t)-\mathbf{x}(v)\|}\right)$$

Proof Consider the shaded cone subtended by the source s, tangent to the circle of radius Δ_Q centered on v. Since the deviation of the default paths is Δ_Q, the intermediate node must lie within the shaded cone if the path $q(s, w)$ is to pass through v. If the intermediate node is in the cone, the random intermediate point $\mathbf{y}$ must lie either in the cone or in one of the two shaded strips of thickness R around the cone. Since $\mathbf{y}$ must also be on $\ell^{\perp}$, $\mathbf{y}$ must lie on the line segment illustrated by the thick line of length ε illustrated in Fig. 13.4. The probability of using v is then bounded by $\varepsilon/\left|\ell_{\mathcal{A}}^{\perp}\right|$. We use the definitions of θ, ϕ as shown in Fig. 13.4. Using some elementary geometry, we find that

$$\varepsilon = R\cdot\left(\frac{1}{\cos(\theta)}+\frac{1}{\cos(\theta+\phi)}\right)+\frac{1}{2}|\ell|\cdot(\tan(\theta+\phi)-\tan(\theta))$$

We observe that ε is largest when $\theta \le \frac{\pi}{4}$ and $\theta+\phi \le \frac{\pi}{4}$, so using some trigonometric identities we get

$$\varepsilon \le 2\sqrt{2}R + \frac{|\ell|}{2}\cdot\frac{\tan\phi(1+\tan^2\theta)}{1-\tan\theta\tan\phi}$$
$$\le 2\sqrt{2}R + |\ell|\cdot\frac{\tan\phi}{1-\tan\phi}$$

where the last line follows because $\tan\theta < 1$. Since $\left|\ell_{\mathcal{A}}^{\perp}\right| \ge \gamma|\ell^{\perp}| = \gamma|\ell|$, we get that the probability of using v is at most

$$\begin{aligned}
\text{Prob} &\leq 2\sqrt{2}\frac{R}{\gamma|\ell|} + \frac{1}{\gamma}\frac{\tan\phi}{1-\tan\phi} \\
&\overset{(a)}{\leq} 2\sqrt{2}\frac{R}{\gamma|\ell|} + \frac{2\tan\phi}{\gamma} \\
&\overset{(b)}{=} 2\sqrt{2}\frac{R}{\gamma|\ell|} + \frac{4}{\gamma}\frac{\tan\frac{\phi}{2}}{1-\tan^2\frac{\phi}{2}} \\
&\overset{(c)}{\leq} 2\sqrt{2}\frac{R}{\gamma|\ell|} + \frac{64\tan\frac{\phi}{2}}{15\gamma} \\
&\overset{(d)}{=} 2\sqrt{2}\frac{R}{\gamma|\ell|} + \frac{64}{15\gamma}\frac{\Delta_Q/r}{\sqrt{1-\Delta_Q^2/r^2}} \\
&\overset{(e)}{\leq} \frac{5}{\gamma}\left(\frac{R}{|\ell|} + \frac{\Delta_Q}{r}\right)
\end{aligned}$$

Inequality (a) follows because when $\tan\phi \leq \frac{1}{2}$, $\tan\phi/(1-\tan\phi) \leq 2\tan\phi$, and when $\tan\phi > \frac{1}{2}$, $2\tan\phi > 1$, in which case it is a trivially valid upper bound for the probability; (b) follows by using a double angle identity; (c) follows by a similar argument that leads to (a) by considering separately $\tan\frac{\phi}{2} \leq \frac{1}{4}$ and $\tan\frac{\phi}{2} > \frac{1}{4}$; (d) follows because from Fig. 13.4, we see that $\tan\frac{\phi}{2} = \Delta_Q/\sqrt{r^2-\Delta_Q^2}$; and finally, (e) follows using $2\sqrt{2} < 5$ and by considering separately the cases $\Delta_Q/r \leq \frac{1}{5}$ and $\Delta_Q/r > \frac{1}{5}$ (similar with (a) and (c)).

To conclude, note that by symmetry, the situation is exactly reversed if the packet uses v in phase II of its path, except that now r will be the distance from v to the destination t. ☒

In order to bound the congestion on node v, we need to bound the number of packets that can cross v and then using Lemma 3 we will be able to bound the expected congestion on v. We first compute how far the packets that cross v have their sources or destinations from v, this will help to bound the number of those packets. Let X^I denote the packets that could possibly use v during phase I of their path and similarly X^{II}. Consider only the packets in X^I. Let $S^I = \{s_k\}$ denote the sources of all the packets in X^I. Let $r_{\max}$ be the maximum (Euclidean) distance from the positions of these sources to $\mathbf{x}(v)$, thus, $r_{\max} = \max_{s\in S^I} \| \mathbf{x}(s) - \mathbf{x}(v) \|$. We have the following result (a similar result holds for the destinations).

Lemma 4 $r_{\max} \leq \frac{D^*}{\sqrt{2}\alpha} + R + \Delta_Q$.

Proof Let s be a source that could possibly use v in phase I and let t be the corresponding destination. Let w be a possible intermediate node. Then $\| \mathbf{x}(s) - \mathbf{x}(w) \| \leq \frac{1}{\sqrt{2}} \| \mathbf{x}(s) - \mathbf{x}(t) \| + R$. Since the path cannot deviate by more than Δ_Q from the line joining $\mathbf{x}(s)$ to $\mathbf{x}(w)$, and the path passes through v, it follows that

$$\begin{aligned}\| \mathbf{x}(v) - \mathbf{x}(s) \| &\leq \| \mathbf{x}(s) - \mathbf{x}(w) \| + \Delta_Q \\ &\leq \frac{1}{\sqrt{2}} \| \mathbf{x}(s) - \mathbf{x}(t) \| + R + \Delta_Q\end{aligned}$$

To conclude, note that $\| \mathbf{x}(s) - \mathbf{x}(t) \| \leq \frac{1}{\alpha}\mathsf{dist}_G(s,t)$ and $\mathsf{dist}_G(s,t) \leq D^*$. ☒

In order to bound the congestion on v, we will divide the area around v into concentric rings with maximum radius $r_{\max}$. We will then bound the number of packets that originate in each ring and use v. The number of packets from each ring will be used to bound the expected congestion caused by each ring. The sum of the expected congestions from the rings will determine the total congestion on node v.

Consider concentric rings $A_0, A_1, A_2, \ldots$ of exponentially increasing radius, centered at $\mathbf{x}(v)$. Ring A_i has radius $r_i = 2^i/\beta$, for $i \geq 0$. Let $i_{\max} = \lceil \log(r_{\max}\beta) \rceil$ (logarithms are base 2). Note that all the sources in S^I are contained in $A_{i_{\max}}$. For $i > 0$, we collect in set S_i^I all the sources which are in ring A_i, but not in A_{i-1} (that is, they are in the area between A_{i-1} and A_i). Figure 13.5 illustrates the situation. Consider a particular i and the packets X_i^I with sources in S_i^I. Let $N_i = \left|X_i^I\right|$ be the number of packets with sources in S_i^I. In order to obtain an upper bound on the expected congestion at v, we will need to bound N_i in terms of the optimal node congestion C^*_{node}.

Lemma 5 *For any* $i \geq 0$:

$$C^*_{\text{node}} \geq \frac{\alpha h_i N_i}{4c(\beta r_i)^2}$$

where,

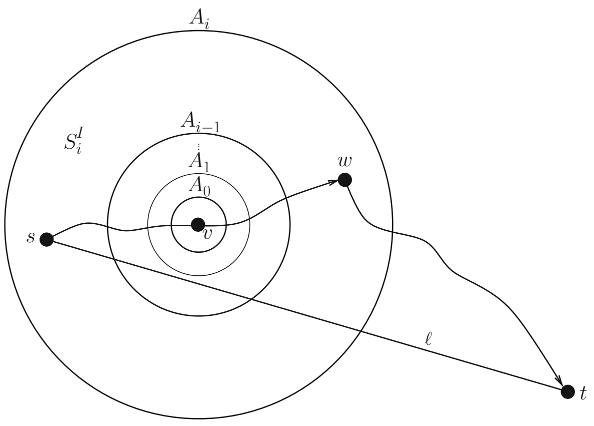


Fig. 13.5 Expected congestion at a node v

$$h_i = \max\left\{\frac{1}{\beta}, \sqrt{2}(r_{i-1} - R - \Delta_Q)\right\}$$

Proof As in the proof of Lemma 4, $\| \mathbf{x}(s) - \mathbf{x}(v) \| \leq \frac{|\ell|}{\sqrt{2}} + R + \Delta_Q$, and since $\| \mathbf{x}(s) - \mathbf{x}(v) \| \geq r_{i-1}$, we get

$$|\ell| \geq \sqrt{2}(r_{i-1} - R - \Delta_Q)$$

From Lemma 1, $|\ell| \geq \frac{1}{\beta}$, therefore $|\ell| \geq h_i$. Furthermore, from the definition of α and Lemma 1, we have that the minimum number of hops from s to t is at least $\alpha|\ell| \geq \alpha h_i$, and each of these hops moves a distance of at most $\frac{1}{\alpha}$. So, for N_i such paths, any path selection algorithm will have to use at least αh_i hops per path, within a disk of radius

$$r = r_i + h_i \leq r_i + 2r_{i-1} \leq 2r_i$$

By Lemma 2, there are at most $4c(\beta r_i)^2$ nodes within this disk of radius r. The minimum total number of times these nodes are used by *any* path selection algorithm is $\alpha h_i N_i$. Thus, the average number of times T_{avg} a node is used in radius r is at least

$$T_{\text{avg}} \geq \frac{\alpha h_i N_i}{4c(\beta r_i)^2}$$

where c is the constant defined in Lemma 2. Since one of these nodes has to be used at least T_{avg} times, we obtain a lower bound on the congestion for any path selection algorithm, and hence for the optimal congestion $C^*_{\text{node}} \geq T_{\text{avg}}$. ⊠

Note that inverting the bound in Lemma 5, we get an upper bound for N_i, when $i \geq 1$:

$$N_i \leq \frac{4c(\beta r_i)^2 C^*_{\text{node}}}{\alpha h_i} \tag{13.1}$$

Note that for $i = 0$ it holds trivially that $N_0 \leq 0$, since no node except for v can be in ring A_0 (a consequence of Lemma 1). The upper bound for N_i together with the upper bound for the probability that any of these packets uses node v (Lemma 3) allows us to bound the expected congestion.

Theorem 2 *The expected congestion on node v is*

$$E[C(v)] \leq f(\gamma, \alpha, \beta, R, \Delta_Q, D^*) \cdot C^*_{\text{node}}$$

where,

$$
\begin{aligned}
&f(\gamma, \alpha, \beta, R, \Delta_Q, D^*) = \\
&\frac{40c\beta^2(R+2\Delta_Q)}{\gamma\alpha} \cdot ((3+4\beta(R+\Delta_Q))^2 \\
&+4\log\left(\frac{\beta D^*}{\sqrt{2}\alpha} + \beta(R+\Delta_Q)\right) + 1\bigg).
\end{aligned}
$$

Proof Let $\text{Prob}_v(\pi)$ be the probability that packet $\pi \in X_i^I$ uses node v. Then packet π's contribution to the expected node congestion at v is $\text{Prob}_v(\pi)$. Using Lemma 3, we can bound $\text{Prob}_v(\pi)$ by P_I. Then, $N_i P_I$ is an upper bound for the contribution to the expected node congestion at v due to the packets in X_i^I. Since every source in S_i^I is distanced at least r_{i-1} from node v, from Lemma 3 and using 13.1, and the fact that $r_i \geq h_i$, we obtain for $i \geq 1$:

$$
\begin{aligned}
\sum_{\pi \in X_i^I} \text{Prob}_v(\pi) &\leq N_i P_I \\
&\leq \frac{20c(\beta r_i)^2 C^*_{\text{node}}}{\gamma\alpha h_i}\left(\frac{R}{h_i} + \frac{\Delta_Q}{r_{i-1}}\right) \\
&= \frac{20c\beta^2 C^*_{\text{node}}}{\gamma\alpha}\left(R\frac{r_i^2}{h_i^2} + 2\Delta_Q\frac{r_i}{h_i}\right) \\
&\leq \frac{20c\beta^2(R+2\Delta_Q)C^*_{\text{node}}}{\gamma\alpha} \cdot \frac{r_i^2}{h_i^2}
\end{aligned}
$$

The expected node congestion at v is obtained by summing the contributions due to each set X_i^I for $i = 1, \ldots, i_{\max}$. Thus,

$$
E[C(v)] \leq \frac{20c\beta^2(R+2\Delta_Q)C^*_{\text{node}}}{\gamma\alpha}\sum_{i=1}^{i_{\max}}\frac{r_i^2}{h_i^2}
$$

Consider now the ratio h_i/r_i. We have

$$
\begin{aligned}
\frac{h_i}{r_i} &= \frac{\max\left\{\frac{1}{\beta}, \sqrt{2}(r_{i-1} - R - \Delta_Q)\right\}}{r_i} \\
&= \max\left\{\frac{1}{2^i}, \sqrt{2}\left(\frac{1}{2} - \frac{\beta(R+\Delta_Q)}{2^i}\right)\right\}
\end{aligned}
$$

Let

$$
i^* = \left\lceil \log\left(\sqrt{2} + 4\beta(R+\Delta_Q)\right)\right\rceil.
$$

Then for $i \geq i^*$, it holds $\frac{h_i}{r_i} \geq \frac{\sqrt{2}}{4}$ or equivalently $\frac{r_i}{h_i} \leq 2^{3/4} < 2$. For $1 \leq i < i^*$, we have that $\frac{h_i}{r_i} \geq \frac{1}{2^i}$, or in other words, $\frac{r_i}{h_i} \leq 2^i$. Since $i_{\max} = \lceil \log(r_{\max}\beta) \rceil$, using the bound in Lemma 4, we get

$$\begin{aligned}
\sum_{i=1}^{i_{\max}} \frac{r_i^2}{h_i^2} &= \sum_{i=1}^{i^*-1} \frac{r_i^2}{h_i^2} + \sum_{i=i^*}^{i_{\max}} \frac{r_i^2}{h_i^2} \\
&\leq \sum_{i=1}^{i^*-1} 4^i + \sum_{i=i^*}^{i_{\max}} 4 \\
&\leq (2^{i^*})^2 + 4 i_{\max} \\
&\leq (3 + 4\beta(R + \Delta_Q))^2 \\
&\quad + 4 \log\left(\frac{\beta D^*}{\sqrt{2}\alpha} + \beta(R + \Delta_Q)\right) + 1
\end{aligned}$$

A symmetrical argument applies to the second phase of the paths, which contributes an additional factor of 2, concluding the proof. ⊠

Note that without increasing the expected congestion, we can always remove any cycles in a path, so without loss of generality, we will assume that the paths are acyclic. We now obtain a concentrated result on the congestion using a straightforward Chernoff bounding argument and the fact that every packet selects its path independently of every other packet. To simplify the presentation, we give the result for constant γ, α, β, R in which case Theorem 2 gives

$$E[C(v)] = O\left(C^*_{\text{node}} \cdot \left(\Delta_Q^3 + (1 + \Delta_Q)\log(D^* + \Delta_Q)\right)\right)$$

The general case can be handled similarly. We have the following theorem.

Theorem 3 *When γ, α, β, R are constants, the node congestion is*

$$C_{\text{node}} = O\left(C^*_{\text{node}} \cdot \left(1 + \Delta_Q^3\right) \cdot \log(n + \Delta_Q)\right)$$

with high probability.

Proof Let $X_i = 1$ if path $p(s_i, t_i)$ uses node v, and $X_i = 0$ otherwise. Then, by Theorem 2, there is a constant A such that

$$\begin{aligned}
E[C(v)] &= E\left[\sum_i X_i\right] \\
&\le A \cdot C^*_{\text{node}} \\
&\quad \cdot \left(\Delta_Q^3 + (1+\Delta_Q)\log(D^* + \Delta_Q)\right) \\
&\le A \cdot C^*_{\text{node}} \\
&\quad \cdot \left(\Delta_Q^3 \log n + (1+\Delta_Q)\log(n(D^* + \Delta_Q))\right) \\
&:= B.
\end{aligned}$$

Let $\kappa > 2e$. Since $\sum_i X_i$ is a sum of independent Bernoulli trials, by applying a Chernoff bound [23] we obtain

$$P[C(v) > \kappa B] < 2^{-\kappa B} \le 1/n^{\kappa A}$$

where we used the facts that $C^*_{\text{node}}, D^* \ge 1$ and $\Delta_Q \ge 0$. Taking a union bound over the n nodes multiplies by an additional n, reducing the exponent on the right to $\kappa A - 1$. Choosing a large enough κ, and noting that $D^* = O(n)$, we obtain the theorem. ☒

13.2.2.3 Edge Congestion Analysis

For the edge congestion, the proof is similar to the node congestion. In order to carry through the same analysis, we need an upper bound on the number of edges in the area, so we can get a lower bound on the average edge congestion. If the maximum degree (maximum number of edges adjacent per node) in the network is δ, then the maximum number of edges is at most a factor of δ times the maximum number of nodes. Therefore, the result is that the optimal edge congestion is at most a factor of δ smaller than the optimal node congestion, giving the following theorem for the expected edge congestion,

Theorem 4 *Let δ be the maximum node degree. The expected congestion on an edge e is*

$$E[C(e)] \le \delta \cdot f(\gamma, \alpha, \beta, R, \Delta_Q, D^*) \cdot C^*_{\text{edge}}$$

A concentrated result can also be obtained for the edge congestion.

Theorem 5 *When γ, α, β, R are constants, the edge congestion is*

$$C_{\text{edge}} = O\left(\delta \cdot C^*_{\text{edge}} \cdot \left(1 + \Delta_Q^3\right) \cdot \log(n + \Delta_Q)\right)$$

with high probability.

13.2.3 Applications of Geometric Networks

The oblivious algorithm for geometric networks has applications in the two-dimensional mesh and also in uniformly distributed unit disk graphs. The two-dimensional mesh is an $\sqrt{n} \times \sqrt{n}$ grid of nodes, where each node is connected with at most four adjacent neighbors (see Fig. 13.2). The nodes are placed at a unit distance from each other, and thus $R = 1/\sqrt{2}$. The rectangular area $\mathcal{A}$ is a square defined by the border nodes of the mesh so the pseudo-convexity $\gamma = 1/2$. For the default path between a pair of nodes, we choose the shortest path that connects the nodes which is closest to the geodesic and therefore $\mathsf{deviation}(Q) \leq 1/\sqrt{2}$. Since the default paths are shortest paths, $\mathsf{stretch}(Q) = 1$. Since adjacent nodes cannot be further than a unit distance, we have that $\alpha = 1$. Moreover, the number of nodes used per unit distance in the shortest path is maximized when the geodesic between the nodes is 45°, which gives $\beta = \sqrt{2}$. Since the maximum node degree is 4, using Theorems 1, 3, and 5, we obtain

Theorem 6 *The oblivious algorithm on the mesh has* $\mathsf{stretch}(P) < 2\sqrt{2}$ *and node congestion* $O\left(C^*_{\text{node}} \cdot \log n\right)$ *and edge congestion* $O\left(C^*_{\text{edge}} \cdot \log n\right)$ *with high probability.*

We consider *uniform disk graphs* with n nodes distributed in an $s_1 \times s_2$ rectangle area $\mathcal{A}$, with constant pseudo-convexity $\gamma = \min\{s_1, s_2\}/2\max\{s_1, s_2\}$ (i.e., the sides are proportional to each other). In a disk graph, each node has a constant radius r and is connected to any node within this radius (see Fig. 13.6). We set the radius $r = 2\sqrt{2}$ and assume that no two nodes are placed within a constant distance l of each other. We consider a uniform distribution for the nodes in the area, i.e., the area is divided into non-overlapping unit squares, and every unit square area contains a number of nodes between 1 and $k = O\left(\frac{1}{l^2}\right)$ nodes, where k is a constant. By the choice of r, two nodes within the same square or in adjacent squares will be connected. Thus, $R \leq \sqrt{2}$, and since there are at most 32 squares containing nodes which could possibly be adjacent to a particular node, the maximum node degree is bounded by $\delta \leq 32k$.

We now explain how to construct the default paths (see Fig. 13.6). Consider two nodes u and v in area $\mathcal{A}$ and construct the line ℓ that connects $\mathbf{x}(u)$ to $\mathbf{x}(v)$. This line passes through a collection of unit squares, forming a path with adjacent unit squares. We pick one node from each square and construct the default path by connecting these nodes. Since for every node in the path, the line passes through the corresponding unit square containing the node, $\mathsf{deviation}(Q) \leq \sqrt{2}$. The number of unit squares in the formation of the default path is no more than $2|\ell|$, so the longest default path consists of at most $2|\ell|$ nodes. The shortest path has to use at least $|\ell|/r$ nodes; therefore, $\mathsf{stretch}(Q) \leq 2r$. Since $\mathsf{dist}_G(u, v) \geq \mathsf{dist}_E(u, v)/r$, $\alpha \geq 1/r$. If $\mathsf{dist}_G(u, v) = 1$, $\mathsf{dist}_E(u, v) \geq l$, so $\mathsf{dist}_G(u, v)/\mathsf{dist}_E(u, v) \leq \frac{1}{l}$. More generally, we know that $\mathsf{dist}_G(u, v) \leq 2|\ell|$ since the default path has $2|\ell|$ hops, so the shortest path cannot have more. Thus, $\mathsf{dist}_G(u, v)/\mathsf{dist}_E(u, v) \leq \max\{2, 1/l\}$, so $\beta \leq \max\{2, 1/l\}$. Applying Theorems 1, 3, and 5, we obtain

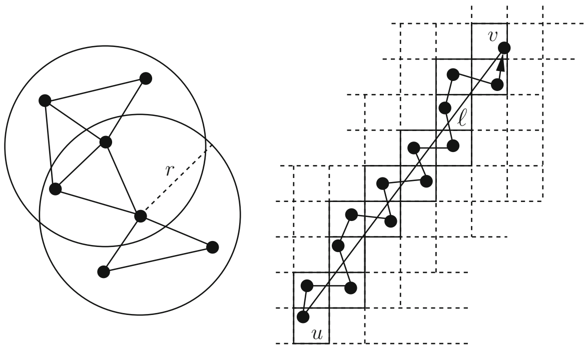


Fig. 13.6 Connectivity of a disk graph and default path construction

Theorem 7 *On uniform disk graphs, the oblivious routing algorithm has* $\mathsf{stretch}(P) = O(1)$ *and node congestion* $O\left(C^*_{\text{node}} \cdot \log n\right)$ *and edge congestion* $O\left(C^*_{\text{edge}} \cdot \log n\right)$ *with a high probability.*

13.3 Mesh Networks

13.3.1 Preliminaries on Mesh Networks

The d-dimensional mesh M is a d-dimensional grid of nodes with side length m_i in dimension i. There is a link connecting a node with each of its $2d$ neighbors (except for the nodes at the boundaries of the mesh). We denote by n the size of M, $n = \mathsf{size}(M) = \prod_{i=1}^{d} m_i$, and by $|E|$ the number of edges in the network. Each node has a coordinate. For example, in the two-dimensional mesh, the top-left node has coordinate $(0, 0)$. We refer to specific submeshes by giving its end points in every dimension, for example, $[0, 3][2, 5]$ refers to a 4×4 submesh, with the x coordinate ranging from 0 to 3 and the y coordinate from 2 to 5.

The input for the path selection problem is a set of N sources and destinations (i.e., packets), $\Pi = \{s_i, t_i\}_{i=1}^{N}$ and the mesh M. The output is a set of paths, $P = \{p_i\}$, where each path $p_i \in P$ is from node s_i to node t_i. The length of path p, denoted $|p|$, is the number of edges it uses. We denote the length of the shortest path from s to t by $\mathsf{dist}(s, t)$. We will denote by D^* the maximum shortest distance, $\max_i \mathsf{dist}(s_i, t_i)$. The *stretch* of a path p_i, denoted $\mathsf{stretch}(p_i)$, is the ratio of the path length to the shortest path length between its source and destination, $\mathsf{stretch}(p_i) = |p_i|/\mathsf{dist}(s_i, t_i)$. The *stretch factor* for the collection of paths P, denoted $\mathsf{stretch}(P)$, is the maximum stretch of any path in P, $\mathsf{stretch}(P) = \max_i \mathsf{stretch}(p_i)$.

For a submesh $M' \subseteq M$, let $\mathsf{out}(M')$ denote the number of edges at the boundary of M', which connect nodes in M' with nodes outside M'. For any routing problem Π, we define the *boundary congestion* as follows. Consider some submesh of the network M'. Let Π' denote the packets (pairs of sources and destinations) in Π which have either their source or their destination in M', but not both. All the packets in Π' will cross the boundary of M'. The paths of these packets will cause congestion at least $|\Pi'|/\mathsf{out}(M')$ times. We define the boundary congestion of M' to be $B(M', \Pi) = |\Pi'|/\mathsf{out}(M')$. For the routing problem Π, the boundary congestion B is the maximum boundary congestion over all its submeshes, i.e., $B = \max_{M' \subseteq M} B(M', \Pi)$. Clearly, $C^* \geq B$.

13.3.2 *Oblivious Routing on Two-Dimensional Mesh Networks*

Here we show how to select the paths in a two-dimensional mesh with equal side lengths $m = 2^k,\ k \geq 0$. We consider this case here for expository ease, however, the result generalizes to the case of unequal side lengths which are not necessarily powers of 2. We use the two-dimensional case to illustrate the main ideas, before generalizing to the d-dimensional case in the next section. The path selection algorithm relies on a decomposition of the mesh to submeshes, and then constructing an access graph, as we describe next.

13.3.2.1 Decomposition to Submeshes

We decompose the mesh M into two types of submeshes, type-1 and type-2, as follows:

- *Type*-1 *submeshes:* We define the type-1 submeshes recursively. There are $k + 1$ levels of type-1 submeshes, $\ell = 0, \ldots, k$. The mesh M itself is the only level 0 submesh. Every submesh at level ℓ can be partitioned into four submeshes by dividing each side by 2. Each resulting submesh is a type-1 submesh at level $\ell + 1$. This construction is illustrated in Fig. 13.7. In general, at level ℓ there are $2^{2\ell}$ submeshes each with side $m_\ell = 2^{k-\ell}$. Note that the level k submeshes are the individual nodes of the mesh.
- *Type*-2 *submeshes:* There are $k - 1$ levels of type-2 submeshes, $\ell = 1, \ldots, k - 1$. The type-2 submeshes at level ℓ are obtained by first extending the grid of type-1 meshes by adding one layer of type-1 meshes along every dimension. The resulting grid is then translated by the vector $-(m_\ell/2, m_\ell/2)$. In this enlarged and translated grid, some of the resulting translated submeshes are entirely within M. These are the *internal* type-2 submeshes. For the remaining *external* type-2 submeshes, we keep only their intersection with M, except that we discard all the "corner" submeshes, because they will be included in the type-1 submeshes at the next level. Notice that all the type-2 submeshes have at least one side with a length of m_ℓ nodes. Fig. 13.7 illustrates the construction.

A submesh of M is *regular* if it is either type-1 or type-2. Unless otherwise stated, a submesh will always refer to regular submeshes. The following lemma follows from the construction of the regular submeshes.

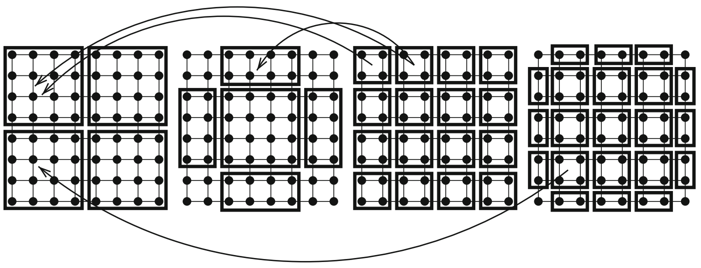


Fig. 13.7 Mesh decomposition for the $2^3 \times 2^3$ mesh. *Arrows* indicate the parents of a submesh

Lemma 6 *The mesh decomposition satisfies the following properties:*

*(1) The type-*1 *submeshes at a given level are disjoint, as are the type-*2 *submeshes.*
*(2) Every regular submesh at level ℓ can be partitioned into type-*1 *submeshes at level $\ell + 1$.*
(3) Every regular submesh at level $\ell + 1$ is completely contained in a submesh at level ℓ of either type-1 or type-2, or both.

13.3.2.2 Access Graph

The access graph $G(M)$, for the mesh M, is a leveled graph with $k + 1$ levels of nodes, $\ell = 0, \ldots, k$. The nodes in the access graph correspond to the distinct regular submeshes. Specifically, every level-ℓ submesh (type-1 or type-2) corresponds to a level ℓ node in $G(M)$. Edges exist only between adjacent levels of the graph. Let u_ℓ, $u_{\ell+1}$ be level ℓ and $\ell + 1$ nodes of $G(M)$, respectively. The edge $(u_\ell, u_{\ell+1})$ exists if the regular submesh corresponding to u_ℓ completely contains the regular submesh corresponding to $u_{\ell+1}$. We borrow some terminology from trees. We say that u_ℓ is a *parent* of $u_{\ell+1}$ in $G(M)$; the parent relationship is illustrated in Fig. 13.7, for the corresponding submeshes. Note that the access graph is not necessarily a tree, since a node can have two parents (a consequence of Lemma 6, part (3)). The depth of a node is the same as its level ℓ, and its height is $k - \ell$. Nodes at height 0 have no children and are referred to as leaves. The leaves in $G(M)$ correspond to single nodes in the mesh. There is a unique root at level 0, which corresponds to the whole mesh M.

Let $p = (u_1, u_2, \ldots, u_k)$ be a path in $G(M)$. We say that p is *monotonic* if every node is of increasing level (i.e., the level of u_i is higher than the level of u_{i+1}), and the respective submeshes of nodes $u_2, \ldots, u_k$ are all of type-1. If p is monotonic, then we say that u_1 is an *ancestor* of u_k. We will use a function g to map nodes in the access graph to submeshes. Let u be a node in the access graph with a corresponding submesh M'. We define the function g so that $g(u) = M'$. Denote by g^{-1} the inverse of function g, that is, $g^{-1}(M') = u$. Using induction on the height of $G(M)$ and part (2) of Lemma 6, we obtain the following lemma:

Lemma 7 *Let v be any node (1×1 submesh) of a regular submesh $M' \subseteq M$, then $g^{-1}(M')$ is an ancestor of $g^{-1}(v)$.*

Let u and v be two leaves of $G(M)$ and let A be their (not necessarily unique) deepest common ancestor; note that A exists and in the worst case is $g^{-1}(M)$ (a consequence of Lemma 7). Let $p = (u, \ldots, A, \ldots, v)$, be the concatenation of two monotonic paths, one from A to u and the other from A to v. We will refer to p as the *bitonic* path between u and v. Submesh $g(A)$ may be type-1 or type-2, all the other submeshes in p are of type-1. We will refer to $g(A)$ as a "bridge" submesh, since it provides the connecting point between two monotonic paths. Note that type-2 submeshes can be used as bridges between type-1 submeshes, when constructing bitonic paths between leaves. Further, only one type-2 submesh is ever needed in a bitonic path. These access graph paths will be used by the path selection algorithm. Suppose that $height(A) = h_A$. The length of a bitonic path from u to v is $2h_A$. We now show that h_A cannot be too large. This will be important in proving that the path selection algorithm gives constant stretch.

Lemma 8 *The deepest common ancestor of two leaves u and v has a height of at most $\lceil \log \mathsf{dist}(g(u), g(v)) \rceil + 2$.*

Proof Let $s, t \in M$ such that $s = g(u)$ and $t = g(v)$. We show that there is a common ancestor with height at most $\lceil \log \mathsf{dist}(s, t) \rceil + 2$.

Assume, first, that instead of a mesh, the network is a torus (the same result holds for the mesh, with a minor technical detail in the proof due to edge effects, which we will discuss later). In this case, all type-2 meshes are of the same size. We obtain the regular submeshes in the original mesh after truncation of the submeshes at the borders of the torus. Note that all distances, however, are measured on the mesh.

Let $\mu = 2^{\lceil \log \mathsf{dist}(s,t) \rceil} \geq \mathsf{dist}(s, t)$. If $4\mu \geq 2^k$, then the root, $g^{-1}(M)$, is a common ancestor with, at most, a height of $\lceil \log \mathsf{dist}(s, t) \rceil + 2$, so assume that $4\mu < 2^k$. Node s is contained in some type-1 submesh of side length 4μ. Without loss of generality (since we are on a torus), assume that this submesh is $M_1 = [0, 4\mu - 1]^2$. If M_1 also contains t, then we are done, since by Lemma 7, $g^{-1}(M_1)$ is a common ancestor at height $\lceil \log \mathsf{dist}(s, t) \rceil + 2$. So suppose that t is contained in some other (adjacent) type-1 submesh M_2. There are two possibilities for M_2.

1. M_1 and M_2 are diagonally adjacent, so without loss of generality, let $M_2 = [4\mu, 8\mu - 1][4\mu, 8\mu - 1]$. Since $\mathsf{dist}(s, t) \leq \mu$, $s \in [3\mu, 4\mu - 1]^2$ and $t \in [4\mu, 5\mu - 1]^2$, and so the type-2 submesh $[2\mu, 6\mu - 1]^2$ contains both s and t.
2. M_2 is laterally adjacent to M_1, so, without loss of generality, let M_2 be to the right of M_1, i.e., $M_2 = [0, 4\mu - 1][4\mu, 8\mu - 1]$. In this case, s must be in the right half of M_1 and t in the left half of M_2, i.e., $s \in [0, 4\mu - 1][3\mu, 4\mu - 1]$ and $t \in [0, 4\mu - 1][4\mu, 5\mu - 1]$. There are four cases:

   a. $s \in [0, 2\mu - 1][3\mu, 4\mu - 1]$ and $t \in [0, 2\mu - 1][4\mu, 5\mu - 1]$, in which case the type-2 submesh $[-2\mu, 2\mu - 1][2\mu, 6\mu - 1]$ contains s, t;
   b. $s \in [2\mu, 4\mu - 1][3\mu, 4\mu - 1]$ and $t \in [2\mu, 4\mu - 1][4\mu, 5\mu - 1]$, in which case the type-2 submesh $[2\mu, 6\mu - 1]^2$ contains s, t;

c. $s \in [\mu, 2\mu - 1][3\mu, 4\mu - 1]$ and $t \in [2\mu, 3\mu - 1][4\mu, 5\mu - 1]$, in which case the type-2 submesh $[\mu, 3\mu - 1][3\mu, 5\mu - 1]$ at height $\lceil \log \mathsf{dist}(s, t) \rceil + 1$ contains s, t;
d. $s \in [2\mu, 3\mu - 1][3\mu, 4\mu - 1]$ and $t \in [\mu, 2\mu - 1][4\mu, 5\mu - 1]$, which is similar to (c).

In all cases, s and t are contained in a submesh of a height of at most $\lceil \log \mathsf{dist}(s, t) \rceil + 2$.

To complete the argument, we now suppose that the network is a mesh (instead of a torus). If the ancestor submesh constructed in the torus is also a regular submesh in the mesh, then there is nothing to prove. So, assume that the ancestor constructed in the torus is not a regular submesh of the mesh. In particular, the ancestor constructed on the torus must be composed of two type-2 submeshes, on opposite sides of the mesh. If s, t are both contained in one of these submeshes, then they are both contained in a type-2 submesh of a height of at most $\lceil \log \mathsf{dist}(s, t) \rceil + 2$. The only remaining case is that s is in one of these submeshes and t is in the other. In this case, $\mathsf{dist}(s, t) \geq 2^{k-1}$, and since $\mu \geq \mathsf{dist}(s, t)$, we have that $\mu \geq 2^{k-1}$, or that $4\mu \geq 2^{k+1}$ which contradicts the assumption that $4\mu < 2^k$, concluding the proof. ⊠

Algorithm 2 Oblivious Routing for two-Dimensional Mesh Networks

```
Input: Source $s$ and destination $t$ in the mesh $M$;
Output: Path $p(s, t)$ from $s$ to $t$ in $M$;
1: Let $(u_0, \ldots, u_l)$ denote a bitonic path in $G(M)$ from $g^{-1}(s)$ to $g^{-1}(t)$;
2: for $i = 0$ to $l$ do
3:     Select a node $v_i$ in $g(u_i)$ uniformly at random; //$v_0 = s$ and $v_l = t$
4:     if $1 \leq i \leq l$ then
5:         Construct subpath $r_i$ from $v_{i-1}$ to $v_i$ by picking a dimension by dimension shortest path[1]
           (an at most one-bend path), according to a random ordering of the dimensions;
6:     end if
7: end for
8: The path $p(s, t)$ is obtained by concatenating the subpaths $r_i$, $p(s, t) = r_0 r_1 \cdots r_{l-1}$;
```

13.3.2.3 Path Selection

Given the access graph, the procedure to determine a path from a given source s to a destination t is summarized in Algorithm 2. Note that the algorithm is oblivious and local, since each source–destination pair can obtain a path independently of the other paths. We will now show that our algorithm with the generalized access graph, in addition to obtaining optimal congestion, also controls the stretch. First we show the constant stretch property of the selected paths.

Theorem 8 *For any two distinct nodes s, t,* $\mathsf{stretch}(p(s, t)) \leq 64$.

Proof Let h be the height of the deepest common ancestor of s and t. Then $p(s, t)$ is the concatenation of paths constructed from the dimension by dimension paths in meshes of sides $2^1, \ldots, 2^{h-1}, 2^h, 2^{h-1}, \ldots, 2^1$. A path in a mesh of side ℓ has length of at most $2\ell - 1$, so by adding the lengths of these paths, we have that

$|p(s,t)| \leq 2(2^1 + \cdots + 2^h + 2^h + \cdots + 2^1 - 2h)$ which implies that $|p(s,t)| \leq 2^{h+3} - 4h$. Since s and t are distinct, $h \geq 1$. By Lemma 8, $h \leq \log \mathsf{dist}(s,t) + 3$, and the theorem follows. ⊠

We now relate the congestion of the paths selected to the optimal congestion C^*. Let e denote an edge in M. Let $C(e)$ denote the load on e, i.e., the number of times that edge e is used by the paths of all the packets. We will get an upper bound on $E[C(e)]$, and then, using a Chernoff bound, we will obtain a concentrated result.

We start by bounding the probability that some particular subpath formed by the path selection algorithm uses edge e. Consider the formation of a subpath r_i from a submesh M_1 to a submesh M_2, such that M_2 completely contains M_1, and e is a member of M_2. According to the path selection algorithm, mesh M_1 is of type-1, thus all of its sides are equal to m_ℓ, where ℓ is the level of M_1. We show the following lemma.

Lemma 9 *Subpath r_i uses edge e with probability at most $2/m_\ell$.*

Proof For subpath r_i, let v_1 be the starting node in M_1 and v_2 the ending node in M_2. Suppose $e = (v_3, v_4)$. Without loss of generality, suppose e is vertical. Since the subpath is a one-bend path, edge e can be used only when either v_1 or v_2 have the same x coordinate as e. This event occurs at a probability of at most $2/m_\ell$. ⊠

Let P' be the set of paths that go from M_1 to M_2 or vice versa. Let $C'(e)$ denote the congestion that the packets P' cause on e. We show

Lemma 10 $E[C'(e)] \leq 2|P'|/m_\ell$.

Proof We can write $P' = P_1 \cup P_2$, where P_1 is the set of subpaths from M_1 to M_2 and P_2 is the subpaths from M_2 to M_1. Then, from Lemma 9, the expected congestion on edge e due to the subpaths in P_1 is bounded by $2|P_1|/m_\ell$. Using a similar analysis, the expected congestion on e due to subpaths in P_2 is bounded by $2|P_2|/m_\ell$. Since the congestion on e due to the paths in P' is the sum of the congestions due to P_1 and P_2, we obtain $E[C'(e)] \leq 2(|P_1| + |P_2|)/m_\ell = 2|P'|/m_\ell$. ⊠

From the definition of the boundary congestion, we have that $B \geq B(M_1, \Pi) \geq |P'|/\mathsf{out}(M_1)$. Therefore, $C^* \geq |P'|/\mathsf{out}(M_1)$. Since each side of M_1 has m_ℓ nodes, we have that $\mathsf{out}(M_1) \leq 4m_\ell$. From Lemma 10, we therefore obtain

Lemma 11 $E[C'(e)] \leq 8C^*$.

We "charge" this congestion to submesh M_2. By Lemma 8, only submeshes up to height $h < \log D^* + 3$ can contribute to the congestion on edge e (submeshes of type-1). By summing the congestions due to these at most $2(\log D^* + 3)$ submeshes (a type-1 and a type-2 submesh at each level), and by using Lemma 11, we arrive at an upper bound for the expected congestion on edge e.

Lemma 12 $E[C(e)] \leq 16C^*(\log D^* + 3)$.

Note that without increasing the expected congestion, we can always remove any cycles in a path, so without loss of generality, we will assume that the paths obtained are acyclic. We now obtain a concentrated result on the congestion C obtained by our algorithm, using the fact that every packet selects its path independently of every other packet.

Theorem 9 *$C = O(C^* \log n)$ with high probability.*

Proof Let $X_i = 1$ if path p_i uses edge e, and 0 otherwise. Then $E[C(e)] = E[\sum_i X_i] \leq 16C^*(\log D^* + 3)$. Let $|E|$ be the number of edges in the mesh. For $|E| > 8$, $E[C(e)] \leq 16C^* \log(|E|D^*)$. Let $\kappa > 2e$, then applying a Chernoff bound [23], and using the fact that $C^* \geq 1$ we find that $P[C(e) > 16\kappa C^* \log(|E|D^*)] < (|E|D^*)^{-16\kappa}$. Taking a union bound over all the edges, we obtain

$$P\left[\max_{e \in E} C(e) > 16\kappa C^* \log(|E|D^*)\right] < \frac{1}{(|E|D^*)^{16\kappa-1}}$$

Using the fact that $D^* = O(|E|)$, $|E| = O(n^2)$, and choosing $\kappa = 2e + 1$, we get $C = O(C^* \log n)$ with high probability. ⊠

Acknowledgments We are grateful to the reviewers of this book chapter.

References

1. J. Aspens, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. Online load balancing with applications to machine scheduling and virtual circuit routing. In: *Proceedings of the 25th ACM Symposium on Theory of Computing*, pages 623–631, ACM Press, San Diego, California, USA, 1993.
2. F. Meyer auf der Heide, C. Schindelhauer, K. Volbert, and M. Grünewald. Congestion, dilation, and energy in radio networks. *Theory of Computing Systems*, 37(3):343–370, 2004.
3. B. Awerbuch and Y. Azar. Local optimization of global objectives: Competitive distributed deadlock resolution and resource allocation. In: *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, pages 240–249, Santa Fe, NM, 1994.
4. Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Racke. Optimal oblivious routing in polynomial time. In: *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, San Diego, CA, ACM Press. pages 383–388, June 2003.
5. M. Bienkowski, M. Korzeniowski, and H. Räcke. A practical algorithm for constructing oblivious routing schemes. In: *Proceedings of the 15th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 24–33, ACM Press, San Diego, California, USA, June 2003.
6. A. Borodin and J. E. Hopcroft. Routing, merging, and sorting on parallel models of computation. *Journal of Computer and System Science*, 30:130–145, 1985.
7. C. Busch, M. Magdon-Ismail, and J. Xi. Oblivious routing on geometric networks. In: *Proceedings of the 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, Las Vegas, NV, pages 316–324, July 2005.
8. C. Busch, M. Magdon-Ismail, and J. Xi. Optimal oblivious path selection on the mesh. *IEEE Transactions on Computers*, 57(5):660–671, May 2008.
9. I. Chatzigiannakis, T. Dimitriou, S. Nikoletseas, and P. Spirakis. A probabilistic algorithm for efficient and robust data propagation in wireless sensor networks. *Ad Hoc Networks*, 4(5): 621 – 635, 2006.

10. I. Chatzigiannakis, S. Nikoletseas, and P. G. Spirakis. Efficient and robust protocols for local detection and propagation in smart dust networks. *Mobile Networks and Applications*, 10(1–2):133–149, 2005.
11. S. Dolev, T. Herman, and L. Lahiani. Polygonal broadcast, secret maturity, and the firing sensors. *Ad Hoc Networks*, 4(4):447 – 486, 2006.
12. S. Dolev and N. Tzachar. Empire of colonies: Self-stabilizing and self-organizing distributed algorithm. *Theoretical Computer Science*, 410(6–7):514–532, 2009.
13. C. Efthymiou, S. Nikoletseas, and J. Rolim. Energy balanced data propagation in wireless sensor networks. *Wireless Networks*, 12(6):691–707, 2006.
14. J. Gao and L. Zhang. Tradeoffs between stretch factor and load balancing ratio in routing on growth restricted graphs. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, New York, NY, pages 189–196, 2004.
15. C. Harrelson, K. Hildrum, and S. Rao. A polynomial-time tree decomposition to minimize congestion. In: *Proceedings of the 15th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 34–43, June 2003.
16. C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transmission Network*, 11(1):2–16, 2003.
17. C. Kaklamanis, D. Krizanc, and T. Tsantilas. Tight bounds for oblivious routing in the hypercube. In: *Proceedings of 2nd IEEE Symposium on Parallel and Distributed Processing (2nd SPAA 90)*, pages 31–36, Crete, Greece, July 1990.
18. F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-scheduling in $O(congestion + dilation)$ steps. *Combinatorica*, 14:167–186, 1994.
19. F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays - Trees - Hypercubes*. Morgan Kaufmann, San Mateo, CA, 1992.
20. T. Leighton, B. Maggs, and A. W. Richa. Fast algorithms for finding O(congestion + dilation) packet routing schedules. *Combinatorica*, 19:375–401, 1999.
21. B. M. Maggs, F. Meyer auf der Heide, B. Vöcking, and M. Westerman. Exploiting locality in data management in systems of limited bandwidth. In: *Proceedings of the 38th Annual Symposium on the Foundations of Computer Science*, pages 284–293, IEEE, Miami Beach, Florida, USA, 1997.
22. F. Meyer auf der Heide and Berthold Vöcking. Shortest-path routing in arbitrary networks. *Journal of Algorithms*, 31(1):105–131, April 1999.
23. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, 2000.
24. R. Ostrovsky and Y. Rabani. Universal $O(\text{congestion+dilation+log}^{1+\varepsilon} N)$ local control packet switching algorithms. In: *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, New York, NY, pages 644–653, May 1997.
25. L. Popa, A. Rostamizadeh, R. Karp, C. Papadimitriou, and I. Stoica. Balancing traffic load in wireless networks with curveball routing. In: *MobiHoc*, 2007.
26. H. Räcke. Minimizing congestion in general networks. In: *Proceedings of the 43rd Annual Symposium on the Foundations of Computer Science*, pages 43–52, IEEE, Vancouver, Canada, November 2002.
27. H. Räcke. *Data management and routing in general networks*. Phd thesis, University of Paderborn, Paderborn, Germany, 2003.
28. H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 255–264, ACM Press, Victoria, British Columbia, Canada, May 2008.
29. P. Raghavan and C. D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.
30. C. Scheideler. Course notes. http://www14.in.tum.de/lehre/2005WS/na/index.html.en.
31. A. Srinivasan and C-P. Teo. A constant factor approximation algorithm for packet routing, and balancing local vs. global criteria. In: *Proceedings of the ACM Symposium on the Theory of Computing (STOC)*, pages 636–643, ACM Press, El Paso, Texas, USA, 1997.

32. L. G. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11:350–361, 1982.
33. L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In: *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, pages 263–277, Milwaukee, Wisconsin, USA, May 1981.
34. F. Zhao and L. J. Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann, San Francisco, CA, USA, 2004.

Chapter 14
Scheduling Algorithms for Tree-Based Data Collection in Wireless Sensor Networks

Ozlem Durmaz Incel, Amitabha Ghosh, and Bhaskar Krishnamachari

Abstract Data collection is a fundamental operation in wireless sensor networks (WSN) where sensor nodes measure attributes about a phenomenon of interest and transmit their readings to a common base station. In this chapter, we survey contention-free *time division multiple access* (TDMA)-based scheduling protocols for such data collection applications over tree-based routing topologies. We classify the algorithms according to their common design objectives, identifying the following four as the most fundamental and most studied with respect to data collection in WSNs: (i) minimizing schedule length, (ii) minimizing latency, (iii) minimizing energy consumption, and (iv) maximizing fairness. We also describe the pros and cons of the underlying design constraints and assumptions and provide a taxonomy according to these metrics. Finally, we discuss some open problems together with future research directions.

14.1 Introduction

Data collection from a set of sensors to a common sink over a tree-based routing topology is a fundamental traffic pattern in wireless sensor networks (WSNs). This *many-to-one* communication pattern in which data flows from many nodes to a single node is known as *convergecast*. One may view convergecast as opposite to broadcast or multicast in which data flows from a single node to a set of nodes in the network. Figure 14.1 shows a simple example that illustrates the characteristics of a typical broadcast and convergecast. In broadcast, as shown in Fig. 14.1a, node s is the message source and nodes a, b, and c are expected recipients. Node a hears the message directly from s and forwards a copy to nodes b and c. In case of a convergecast, as shown in Fig. 14.1b, nodes a, b, and c each has a message destined to the sink node s and a serves as a relay for b and c.

O. Durmaz Incel (✉)
NETLAB, Department of Computer Engineering, Bogazici University, 34342 Bebek, Istanbul, Turkey
e-mail: ozlem.durmaz@tam.boun.edu.tr

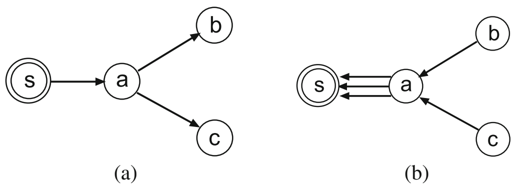


Fig. 14.1 (**a**) Broadcast — data flows from a single node s to a set of nodes a, b, c. (**b**) Convergecast — data flows from nodes a, b, and c to a single node s

Once data is collected at the sink, it either can be recorded and stored for future analysis or can be processed immediately to take certain actions depending on application requirements. In a WSN, data collection either can be triggered by external sources, such as *queries* to get a snapshot view of the network or *events* as and when they appear, or can be for continuous periodic monitoring without any external triggering. In all cases, however, the many-to-one communication pattern is common.

Depending on application requirements, different objectives can be associated with data collection. For instance, in disaster early warning applications, such as detection of forest fire [73] and gas/oil leaks [14], or structural damage identification [9], bursty traffic generated by events needs to be delivered to the sink as quickly and as reliably as possible to prevent catastrophes. On the other hand, in applications where sensor nodes only report periodic data, such as animal habitat monitoring [50], energy efficiency may become a more important concern as opposed to quick data collection.

Particularly under regular, heavy traffic conditions, contention-free medium access control (MAC) protocols, such as *time division multiple access* (TDMA), where nodes communicate on different time slots to prevent conflicts, offer several advantages for data collection as compared to contention-based protocols [27]. They eliminate collisions, overhearing, and idle listening, which are the main sources of energy consumption in wireless communications [15]. In addition, they also permit nodes to enter into sleep modes during inactive periods, thus achieving low duty cycles and conserving energy. Furthermore, TDMA-based communications can provide provable guarantee on the completion time of data collection, for instance, in timely detection of events. Another key aspect of time-slotted communication is robustness during peak loads. When the number of source nodes are many or the data rates are high, carrier-sense multiple access protocols, such as CSMA, may fail to successfully allocate the medium, causing retransmissions and collisions. A number of TDMA-based MAC protocols for WSNs have been proposed to exploit these advantages [2, 20, 36, 58, 61].

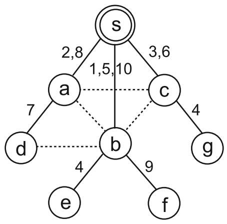
In this chapter, we survey TDMA-based scheduling algorithms for data collection in sensor networks. We first classify the algorithms based on their common objectives and then identify different design constraints and assumptions and provide a taxonomy according to these metrics. We identify the following four objectives as the most studied in literature and the most fundamental with respect to data collection in WSN: (i) minimizing schedule length, (ii) minimizing latency,

(iii) minimizing energy consumption, and (iv) maximizing fairness. We also find that some algorithms target joint optimization of multiple objectives, such as minimizing delay as well as energy. We note that there exist many other studies, not necessarily about convergecast, that focus on these objectives; however, in this chapter, we consider only studies that use TDMA scheduling for tree-based data collection.

In terms of design constraints and assumptions, the algorithms differ mainly in the following dimensions: (a) use of communication and interference models, (b) implementation methods, such as centralized or distributed, (c) topology assumptions, (d) types of data collection, such as use of in-network processing versus raw data, and (e) capability of transceivers available on the sensor nodes. We briefly explain the fundamentals of the algorithms, give the details of some of the highlighted ones, discuss the advantages and disadvantages of the algorithms, and comment on the pros and cons of the design constraints and assumptions. For instance, most of the scheduling algorithms use the *protocol model* [34] for interference, which is a graph theoretic approach that assumes correct reception of a message if and only if there is no other simultaneous transmission within proximity of the receiver. Although this enables the use of simple graph coloring-based scheduling schemes, the model is idealistic and may fail in practice, because interference is not a binary phenomenon, and two nearby concurrent transmissions can actually be successful if the interference level is tolerable. To this end, the use of the *physical model*, which is based on the *signal-to-interference-plus-noise-ratio* (SINR), provides a better solution in terms of realistic capturing of interference from multiple transmissions, thus resulting in correct and realizable schedules [33, 38].

Due to its ability to provide time bounds, TDMA-based scheduling algorithms are widely exploited for *fast* and *timely* delivery of data with the objective of minimizing the time to complete convergecast, i.e., minimizing the latency. In a TDMA schedule, time is slotted and each slot is long enough for transmission or reception of a single packet. Consecutive time slots are grouped into non-overlapping frames, and the schedule for each frame is repeated when data collection is periodic. It is assumed that some form of time synchronization exists among the nodes, which can be achieved using one of the protocols such as [19]. Under this setting, minimizing the data collection time for (aggregated/raw-data) convergecast is equivalent to minimizing the number of time slots required per frame, called the *schedule length*, such that all (aggregated/raw) packets from the source nodes reach the sink.

Since multi-hop TDMA allows spatial reuse of time slots, more than one node can transmit simultaneously if their receivers are in non-conflicting parts of the network. There are two types of conflicts that arise: (i) *primary conflict* and (ii) *secondary conflict*. A primary conflict occurs when a node transmits and receives at the same time or receives more than one transmissions destined to it at the same time. This is due to the single, half-duplex transceiver on each node, which is typical of current WSN hardware [4, 62]. A secondary conflict occurs when a node, an intended receiver of a particular transmission, is also within the range of another transmission intended for other nodes. For instance, in Fig. 14.2, nodes b and d cannot be scheduled simultaneously under the protocol interference model because b's transmission, which is intended for sink s, will interfere with d's transmission at



	Frame									
slots	1	2	3	4	5	6	7	8	9	10
nodes	b	a	c	e,g	b	c	d	a	f	b

Fig. 14.2 Raw-data convergecast that takes 10 time slots to complete. Table shows the schedule

a due to the presence of the interfering link, as indicated by the dotted line between *b* and *a*. For the same reason, nodes *d* and *e* and nodes *a* and *b* cannot be scheduled simultaneously. The figure also illustrates raw-data convergecast where the routing tree rooted at sink *s* is indicated by the solid lines. The table on the right shows the schedule for one frame with the minimum possible schedule length of 10 slots.

The rest of the chapter is organized as follows: Sect. 14.2 explains the details of the followed classification methodology. Section 14.3 presents the existing work on scheduling algorithms for data collection with different objectives and comparisons, along with a taxonomy. Section 14.4 discusses open problems and future research directions. Section 14.5 draws some relevant conclusions.

14.2 Classification Approach and Methodology

Our surveyed algorithms have the common purpose of "*data collection using TDMA-based communication schedules in WSNs*." However, they differ according to their objectives, which are usually set by varying application requirements and the underlying design constraints and assumptions. Our classification methodology is primarily based on the design objectives. In this section, we summarize the most studied objectives, describe their importance in the context of data collection applications, explain the underlying design constraints and assumptions, and comment on the pros and cons of alternative models or approaches that can be used.

14.2.1 Design Objectives

1. ***Minimizing Schedule Length***: Minimizing the schedule length or, equivalently, minimizing the time to complete convergecast, is the most studied design objective for data collection in sensor networks. It translates to quicker data collection at a fast rate. In many WSN applications, it is of interest to maximize the rate at which the sink can receive data from the network [52]. For instance, it is noted that in networked structural health monitoring, more than 500 samples per second are required to efficiently detect and localize damages [9]. Second, a minimal length TDMA schedule allows for a longer sleep period in each data collection cycle, especially for periodic traffic, which contributes to lesser energy

consumption on the sensor nodes. Minimal schedule length can be achieved by maximizing the reuse of the time slots. Therefore, most of the existing algorithms aim to maximize the number of concurrent transmissions and enable spatial reuse by devising strategies to eliminate interference. In certain cases, a minimal schedule length may also contribute to minimizing the latency of data collection. However, the transmission sequence and the number of hops to reach the sink should also be jointly considered as factors impacting the latency.

2. ***Minimizing Latency***: Minimizing the data collection latency is important for applications that are required to take certain (precautionary) actions based on deadlines, such as mission-critical and event-based applications. Although minimizing the schedule length may as well minimize the latency under certain conditions, most algorithms do not consider the *average* latency experienced by individual packets at each hop. Moreover, the ones that aim to minimize the schedule length achieve this by maximizing the reuse of time slots, which, however, for some topologies does not necessarily result in schedules with minimal delay. For instance, a line topology may allow higher spatial reuse, but due to larger number of hops from the sources to the sink, it may cause high latency. Therefore, minimizing the data collection latency may require considering additional constraints in addition to those required for minimizing the schedule length.
3. ***Minimizing Energy Consumption***: Minimizing energy consumption and maximizing network lifetime are fundamental to successfully operating resource-constrained WSNs for long durations. As a major source for energy consumption, radio activity should be managed efficiently. The most common method is to operate the radio in duty cycles with periodic switching between sleep and wake-up modes, which can be easily incorporated with TDMA schedules by maximizing the sleep time. Transmission power control is another well-known technique to reduce energy consumption, contention, and packet losses. Instead of transmitting at maximum power, sending packets at an optimal power level can save energy and extend network lifetime. It is well known that wireless links exhibit variable link qualities over time (due to multi-path effects, fading, etc.), and transmission power control can transform bad quality links into good ones with high packet delivery rates. In addition, packet losses due to contention during peak traffic periods can also be mitigated by power control.
4. ***Maximizing Capacity***: Although maximizing the throughput capacity is not considered to be one of the primary objectives in low-rate data collection applications over small networks, it is important for large, dense sensor networks and for complex applications that require efficient delivery of large amounts of data. The performance of a data-gathering WSN can be characterized by the rate at which information can be delivered to the sink [52]. However, maximizing capacity and minimizing energy consumption are conflicting to each other, and so studying their trade-off is an interesting topic for complex data-gathering applications.
5. ***Maximizing Fairness***: Fairness is one of the key objectives in WSN applications in order to maintain a balanced view of the sensor environment. In applications where each of the sensor readings is important, fairness becomes an important

issue, especially under high data rates. For instance, fair data gathering may become necessary for reducing the estimation error in an application involving field reconstruction.

6. ***Other Objectives***: Minimizing communication costs, maximizing parallel transmissions, meeting deadlines, minimizing interference, and self-stabilization are some of the other objectives studied for data collection in WSNs.
7. ***Joint Objectives***: In most applications, there is not always a single objective, but often multiple, and sometimes conflicting, objectives involved. Some examples include minimizing communication cost and delay, minimizing energy consumption and completion time of data collection, maximizing capacity and minimizing energy. Scheduling algorithms need to address the optimal trade-offs in satisfying these conflicting objectives.

14.2.2 Design Constraints and Assumptions

In this section, we discuss the underlying design constraints and assumptions that the algorithms are based on, ranging from communication models to hardware issues.

1. ***Communication and Interference Models***: In the literature, there are two common approaches to model interference: (i) *protocol model* and (ii) *physical model* [34], also known as the SINR model. The protocol model states that a message is correctly received if there is no other sender transmitting at the same time within a close proximity of the intended receiver. The advantage of this approach is that it enables the use of simple graph coloring-based scheduling algorithms. In [33], Gronkvist et al. analyze the performance of the protocol interference model and indicate that it does not always provide a comprehensive view of reality due to the cumulative effects of interference in wireless networks. The model can also be pessimistic at times, such as when two nearby communications, which are concurrently not admissible by protocol model constraints, can actually take place if the interference level is tolerable. Other models, such as those based on RTS/CTS or hop counts, are extensions or special cases of the protocol model. The physical model, on the other hand, is richer in the sense that it can capture cumulative interference from multiple concurrent transmissions and considers a message to be successfully received if the SINR at the receiver is greater than a certain threshold. Moscibroda [52] studies the impact of the physical model on the achievable capacity in wireless multi-hop networks and shows that protocols designed with the SINR model can surpass the theoretically achievable performance of graph-based scheduling protocols.
2. ***Types of Data Collection***: We identify two fundamental types of data collection in WSN: (i) *raw-data convergecast*, where every packet is relayed individually and (ii) *aggregated convergecast*, where packets are aggregated at each hop before being relayed. Aggregated convergecast is applicable when a strong correlation exists in the sensor readings or the goal is to collect summarized information, such as the maximum sensor reading. Raw-data convergecast, on the other hand, is applicable when every measurement is equally important or

the correlation is minimal. These two types correspond to two extreme cases of data collection in sensor networks.

3. ***Network Topology***: Tree-based routing topologies are most common in many-to-one data collection; however, line, star, or dynamic routing topologies have also been considered. Many of the works assume a fixed topology while some others consider dynamic routing, where the next-hop forwarding node is selected based on some criteria, such as battery level or link reliability.
4. ***Sensor Deployment***: The sensor deployment method usually varies with application requirements. Besides the commonly used random and grid deployments, some applications may support redeployment of nodes, for instance, to eliminate sensing holes in the network [29].
5. ***Buffer Size***: As nodes generate and forward packets toward the sink, they may need to buffer the packets before transmissions. Although some algorithms assume unlimited buffer sizes, minimizing the buffer size can offer advantages considering the limited memory resources available at the nodes.
6. ***Transceiver Assumptions***: Each sensor node is typically equipped with a single, omnidirectional radio that can be tuned to a single channel at any given time. However, radios with multiple transceivers that can support multiple channels and directional antennas are also studied in the literature.
7. ***Implementation Method***: While some algorithms rely on the sink node to compute schedules in a centralized way, others take a distributed approach where nodes compute their schedules based on local information exchanged in their neighborhood.
8. ***Use of Joint Solutions***: Use of TDMA scheduling together with transmission power control, multi-channel scheduling, specific routing solutions are common approaches in the surveyed algorithms.
9. ***Data Collection Pattern***: Data collection can be periodic or one shot. One-shot data collection is typically query based that provides a snapshot of the monitored conditions or event based where nodes report data if an event of interest occurs.
10. ***Granularity of Assignments***: There are two general methods in TDMA scheduling: (i) *node scheduling*, also referred to as broadcast scheduling, and (ii) *link scheduling*, also known as link activation or point-to-point scheduling. In broadcast scheduling, time slots are assigned to the nodes, and a node's transmission is intended for all its neighbors [59]. In link scheduling, individual links are scheduled such that a transmission is intended for and must be received collision free by a particular neighbor. Most of the algorithms use link scheduling in convergecast.

14.3 Scheduling Algorithms for Data Collection

14.3.1 Algorithms on Minimizing Schedule Length

In this section, we survey the TDMA-based scheduling algorithms that identify minimizing the schedule length as their primary objective. We first describe works that pertain to raw-data convergecast and then focus on aggregated convergecast. Since packets are aggregated at each hop in aggregated convergecast, the number of packets transmitted and delivered to the sink is substantially lower than that of raw-data convergecast.

14.3.1.1 Raw-Data Convergecast

For raw-data convergecast, finding a minimum-length, conflict-free assignment of time slots, such that every packet generated by a node reaches the sink, is fundamental to efficient network operations. Several variants of the problem exist depending on network topology, interference model, packet generation scheme, number of frequency channels, buffer constraints, antenna models, etc.

One of the early works in this category is by Florens et al. [22–24], who address the problem of scheduling for packet *distribution* in sensor networks and argue that it can be considered as an inverse of the convergecast problem. Assuming the protocol interference model, they propose optimal centralized algorithms for special network topologies, such as line, multi-line, and tree networks, for both omnidirectional and directional antennas.

For packet distribution in line networks, where the sink sends $p(i) \geq 0$ packets to node i which is i hops away, the basic idea is to first transmit packets destined to the furthest node, then packets for the second furthest node, and so on, as quickly as possible respecting channel reuse constraints. Nodes between the sink and a packet's destination are required to forward that packet as soon as it arrives (i.e., in the next time slot following its arrival). This basic idea can be extended to a multi-line network and a tree network as well. The upper part of Fig. 14.3 shows an example of scheduling for packet distribution on a 10-node line network for directional antennas with $p(i) = 2$, $p(2) = 1$, $p(8) = 1$, and $p(9) = 1$. Once an optimal schedule is found, the schedule for the inverse problem of convergecast where node i sends $p(i)$ packets to the sink is constructed by symmetry as shown in the bottom part of Fig. 14.3. In particular, a transmission from node i to $i + 1$ occurring at time slot j for the distribution problem corresponds to a transmission from node $i + 1$ to i in time slot $T - j + 1$ for the convergecast problem. Here, N is the total number of nodes and T is the minimal schedule length which for omnidirectional antennas is given by

$$T = \max_{1 \leq i \leq N} \left(i - 1 + p(i) + 2p(i+1) + 3 \sum_{j \geq i+2}^{N} p(j) \right) \tag{14.1}$$

and for directional antennas is given by

$$T = \max_{1 \leq i \leq N-1} \left(i - 1 + p(i) + 2 \sum_{j \geq i+2}^{N} p(j) \right) \tag{14.2}$$

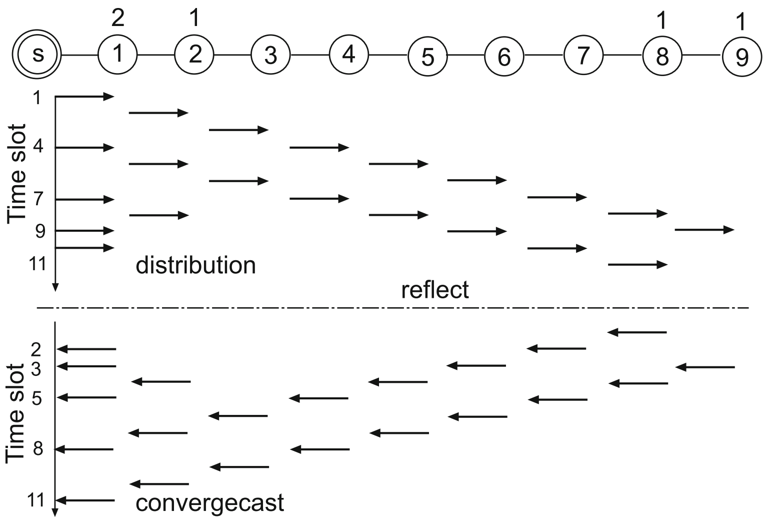


Fig. 14.3 Optimal time scheduling for a 10-node line network with minimum schedule length 11. *Upper part* shows the schedule for packet distribution, and *bottom part* shows the schedule for convergecast, which is obtained by symmetry, i.e., by reflecting the upper schedule with respect to the fictitious *horizontal line*. Note that nodes that are closer than or at two hops do not transmit concurrently to respect interference issue

Ergen et al. in [20] prove that the problem of minimizing the schedule length is NP-complete by reducing it from the *graph coloring* problem. The scheduling difficulty arises since many subsets of non-conflicting nodes are candidates for transmission in each time slot, and the subset chosen in one slot affects the number of transmissions in the next slot. This is due to the fact that some eligible nodes may not have any packet to transmit because of the subset selected in the previous slot. When a graph-based interference model is used, a conflict-free schedule can be found by coloring a *conflict graph*. A conflict graph is one in which every node represents an edge in the original graph and two nodes are connected if their corresponding edges interfere in the original graph, i.e., give rise to primary or secondary conflicts. Using such a graph coloring strategy, they propose a *node-based* and a *level-based* scheduling heuristic and show that one outperforms the other depending on the distribution of packet generation in the network. In particular, node-based scheduling is better for topologies that have equal density of packets across the network or higher density of packets at low levels of the tree, whereas level-based scheduling is better for topologies when the packet density is higher at the upper levels of the tree.

A *virtual node expansion*-based approach that also uses graph coloring to find a minimum-length, conflict-free schedule where every node generates a single packet in each frame is proposed by Lai et al. [41]. They first construct a conflict graph from the original graph and then expands it by creating, for each parent node, a number of virtual nodes equal to the size of the subtree rooted at that node in the original

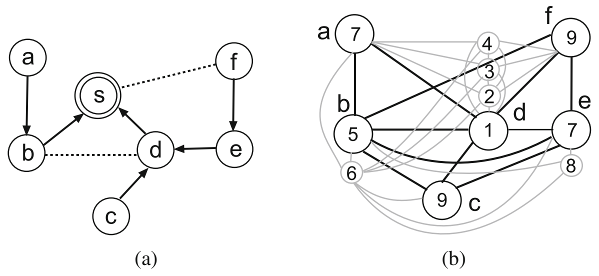


Fig. 14.4 (**a**) The original graph and the routing tree: *arrows* indicate tree edges and *dotted lines* represent interfering links. (**b**) *Dark circles* and *lines* represent nodes and edges in the conflict graph, which is expanded by adding virtual nodes and virtual edges, marked in *gray*

tree. As illustrated in Fig. 14.4b, four virtual nodes, marked as 1, 2, 3, and 4, are created for node d in the *expanded conflict graph* as the size of the subtree rooted at d is four in the original graph, as shown in Fig. 14.4a. This graph expansion is done to accommodate multiple transmissions by intermediate parent nodes which relay packets from nodes in its subtree. Since the virtual nodes also conflict with any node that has an edge to its original node, edges are added between the virtual nodes and the conflicting node. Similarly, an edge is added between each virtual node and its original node.

Once the expanded conflict graph is constructed, an approximate coloring algorithm, originally due to Li et al. [45], is used to find a time slot assignment. The coloring algorithm works by finding a vertex with the least degree and removing it from all its adjacent edges. This is repeated until all the vertices are removed, after which it greedily assigns colors in the reverse order of removing the vertices. This results in a conflict-free schedule for each of the edges in the original graph. As illustrated in Fig. 14.4b, the following schedule is generated: d transmits in slots 1, 2, 3, and 4[1]; b transmits in slots 5 and 6; a transmits in slot 7; e transmits in slots 7 and 8; and c and f both transmit in slot 9. It is shown that the schedule length achieved by this coloring algorithm is no more than $2\delta/k + 1$, where δ is the largest degree in any subgraph of the conflict graph in which every vertex has a degree at least δ and k is the maximum size of an independent set in the neighborhood of any node in the conflict graph.

The authors of this paper also discuss the effect of different routing structures on the schedule length and propose a *disjoint-strip*-based approach to construct an efficient routing topology. This results in uniform flow of data along different paths in the network and prevents certain nodes from being overloaded. The basic idea is

[1] There is no causality constraint, such that node d does not need to wait for data from its children before being scheduled, and the data collection is periodic.

to construct several disjoint, equally spaced *node strips*, all with the same number of nodes. Two distanced strips are likely to relay data simultaneously without interfering with each other. It is shown that this disjoint-strip routing, although increases the total number of transmissions, yields a shorter schedule length for unbalanced node deployments as compared to shortest path routing, which is suitable for balanced deployments minimizing the total number of transmissions but not necessarily the schedule length.

Choi et al. in [12] formulate the scheduling problem as a *minimum information gathering time problem*, where every node has a single packet to send and the goal is to find routing paths from the nodes to the sink as well as an optimal time slot assignment. By reducing it from the *partition problem*, they prove that the problem is NP-complete on general graphs and propose algorithms for line and tree topologies that take at most $3N-3$ time slots to deliver all the packets to the sink. For general networks a heuristic is proposed, which starts with a minimum spanning tree and trims the edges such that transmissions on different branches of the tree do not interfere with each other and can be scheduled in parallel. This results in a backbone forest whose segments are then scheduled independently respecting adjacency and two-hop interfering constraints.

Following a strategy similar to [24], Gandham et al. in [26, 27] propose distributed scheduling algorithms for raw-data convergecast where every node generates a single packet in one data collection cycle. They give an *integer linear programming* (ILP) formulation of the problem and propose a distributed time slot assignment scheme that takes (i) at most $3N-3$ time slots for linear networks, which is optimal, (ii) at most $\max(3n_k-1, N)$ time slots for multi-line and tree networks, where the lower bound for multi-line networks is $\max(3n_k-3, N)$, and (iii) at most $3N$ time slots for general networks. Here n_k represents the maximum number of nodes in any subtree of the routing structure. Similar results are also obtained by Tsai et al. [69]. We note that (14.1) reduces to $3N-3$ when $p_i = 1$ for $i = 1, \ldots, N$, which matches with Gandham's result for line networks.

In addition to minimizing the schedule length, the proposed algorithm in [27] also considers memory constraints on the sensor nodes and requires storage for at most two packets in each node buffer. Links are assumed to be symmetric and the interference model is assumed to be graph based, with the interference range of a node equal to its transmission range. Their results also extend to the case where nodes generate multiple packets and when channel propagation characteristics are not ideal. In the following, we first give their ILP formulation and then briefly explain the details of the algorithm.

Let $G = (V, E)$ represent the sensor network, where V is the set of nodes including the sink s and E is the set of wireless links. Let $p_0(v)$ be the number of packets originated at node v and $p_t(v)$ be the number of packets at node v at the end of slot t. Let $f_t(u, v) \in \{0, 1\}$ represent the state of the link (u, v) at slot t; $f_t(u, v) = 1$ if node u transmits a packet to node v in slot t and 0 otherwise. Let $N(v)$ be the number of one-hop neighbors of node v and T be the number of time slots to complete convergecast. Then the ILP is given by the following:

$$\textbf{Minimize} \quad T$$

$$\text{subject to}:$$

$$p_T(s) = \sum_{u \in V} p_0(u) \tag{14.3}$$

$$\forall u \in V,\ \forall t \in \{1,\ldots,T\}: \quad \sum_{w \in N(v)} \sum_{v \in N(u)} f_t(v,w) \le 1 \tag{14.4}$$

$$\forall u \in V,\ \forall t \in \{1,\ldots,T\}: \quad \sum_{v \in N(u)} f_t(u,v) \le 1 \tag{14.5}$$

$$\forall u \in V,\ \forall t \in \{1,\ldots,T\}: \quad \sum_{v \in N(u)} (f_t(v,u) + f_t(u,v)) \le 1 \tag{14.6}$$

$$\forall u \in V,\ \forall t \in \{1,\ldots,T\}: \quad p_t(u) + \sum_{v \in N(u)} f_t(u,v) = p_{t-1}(u) \tag{14.7}$$

$$\forall u \in V,\ \forall t \in \{1,\ldots,T\}: \quad p_t(u) - \sum_{v \in N(u)} f_t(v,u) = p_{t-1}(u) \tag{14.8}$$

$$\forall u, v \in V: \quad f_t(u,v) \in \{0,1\} \tag{14.9}$$

The objective is to minimize T, the total number of time slots required to complete convergecast. Constraint (14.3) ensures that all packets are delivered to the sink at the end of convergecast. Constraint (14.4) states that at most one neighbor of a node can transmit in a slot, thus addressing the hidden terminal problem. Similarly, constraint (14.5) states that a node transmits to at most one neighbor, and constraint (14.6) restricts a node from both transmitting and receiving in the same time slot. Constraints (14.7) and (14.8) are for conservation of messages, and constraint (14.9) guarantees an integral solution. The above ILP can be solved using tools like CPLEX; however, typically solutions to ILP's have exponential running time. Moreover, such a solution will be centralized and not scalable in nature. Hence combinatorial solutions are preferred.

Starting from linear networks, the algorithm proposed by Gandham et al. is generalized for multi-line networks, which are multiple linear networks intersecting with the sink, and also to tree networks. The basic idea for linear networks is that each node is assigned an initial state depending on its hop count from the sink. As illustrated in Fig. 14.5a, a node at hop distance h is assigned a state of transmitting (T) if h mod 3 is 1, idle (I) if h mod 3 is 2, and receiving (R) if h mod 3 is 0. A node comes back to this initial state after every three time slots and follows the state transition diagram as shown in Fig. 14.5b. Effectively, this implies that for every node in state R, there is only one node in the neighborhood which is in state T, and so packet transmission is always successful resulting in exactly one packet reception by the sink in every three time slots.

The above basic idea extends to more complex topologies, such as multi-line networks, as shown in Fig. 14.5c, tree networks where transmissions are scheduled at parallel along multiple branches, and general networks where packets are routed over a breadth first search (BFS) tree. However, for the distributed algorithm to

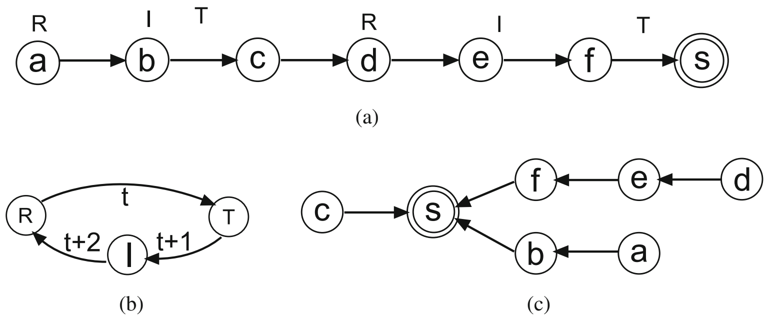


Fig. 14.5 (**a**) A linear network with initial state assignment (R, I, T) depending on the hop distance from the sink s. (**b**) State transition of the nodes. (**c**) A multi-line network as a composition of multiple linear networks

work, each node must know the branch ID and the number of nodes in all the other branches, but need not be aware of the entire network topology. The scheduling rule for multi-line networks is that the branch with the largest number of remaining packets and whose root has at least one packet gets priority to transmit to the sink (ties are broken based on the lowest branch ID). This results in a schedule length of $\max(3n_k - 1, N)$. For general networks, since there are interfering edges that are not part of the spanning tree, the goal is to first eliminate interference by constructing a BFS tree and then scheduling as before. However, in addition to knowing the number of nodes in all the branches and the branch ID, for general networks, a node also has to know a conflict map at the initialization phase. This gives a schedule length of $3N$, although the simulations presented in the paper require only $1.5N$ time slots.

The use of orthogonal codes, such as direct sequence spread spectrum (DSSS) and frequency hopping spread spectrum (FHSS), to eliminate interference is studied by Annamalai et al. [1]. They propose a greedy, top–down tree construction scheme that chooses the children of a node based on the nearest neighbor criterion starting from the sink and traversing the graph in BFS order. To reduce interference, nodes that fall within the transmission range of a parent other than their own are assigned different codes if available; otherwise, the code that is least used by the interfering neighbors is used. Once the channel allocation is done, time slots are assigned in a greedy fashion such that a parent does not transmit before its children. Simulation results indicate that the schedule length on such a tree constructed specifically for convergecast is shorter than on a tree constructed for broadcast [11]. However, one limitation of this approach is that the miniature hardware design of sensor nodes may not permit employing complex radio transceivers required for spread spectrum codes or frequency bands systems.

In [37], Incel et al. explore and evaluate a number of different techniques using realistic simulation models to study the data collection rate for raw-data convergecast. First, a simple spatial-reuse TDMA scheme is employed to minimize the schedule length, which is then combined with multiple frequency channels and transmission power control to achieve further improvement. A receiver-based channel assignment (RBCA) scheme is proposed where the receivers (i.e., parents) of the tree are statically assigned a channel and the children of a common receiver transmit on that channel. This avoids pairwise, per-packet channel negotiation overheads. Once multiple frequencies are used to completely eliminate interference (i.e., secondary conflicts), it is shown that the lower bound on the schedule length is $\max(2n_k - 1, N)$, and a time slot assignment scheme is proposed that achieves this bound with no nodes requiring to buffer more than one packet at any time. Here, as in [27], n_k is the maximum number of nodes in any branch of the tree.

Next, the authors of [37] show that once interference is eliminated, the data collection rate often becomes limited by the routing topology. To overcome this, trees with specific properties are constructed, which help in further enhancing the data collection rate. In particular, *capacitated minimum spanning trees* [57], which aim to have an equal number of nodes on each branch, are shown to achieve a factor of two improvement as compared to single-channel TDMA scheduling on minimum-hop shortest path trees.

Algorithm 1 LOCAL-TIMESLOTASSIGNMENT

```
1.  node.buffer = full
2.  if {node is sink} then
3.          Among the eligible top-subtrees, choose the one with the largest number of total
            (remaining) packets, say top-subtree $i$
4.          Schedule link $(root(i), s)$ respecting interfering constraint
5.  else
6.      if {node.buffer == empty} then
7.              Choose a random child $c$ of node whose buffer is full
8.              Schedule link $(c, node)$ respecting interfering constraint
9.              $c$.buffer = empty
10.             node.buffer = full
11.     end if
12. end if
```

The key idea behind our algorithm, which is formally presented in Algorithm 1 as LOCAL-TIMESLOTASSIGNMENT, is to (i) schedule transmissions in parallel along multiple branches of the tree and (ii) keep the sink busy in receiving packets for as many time slots as possible. Each node maintains a buffer and its associated state, which can be either *full* or *empty* depending on whether it contains a packet or not. Initially, all the buffers are full because every node has a packet to send.

The first block of the algorithm in lines 2–4 gives the scheduling rules between the sink and the roots of the top-subtrees. A *top-subtree* $\text{TS}(r)$ is defined as one whose root r is a child of the sink, and it is said to be *eligible* if r has at least one packet to send. For instance, in Fig. 14.6a, the top-subtrees are $\{1, 4\}$, $\{2, 5, 6\}$, and $\{3, 7\}$. For a given time slot, the root of an eligible top-subtree which has the *largest* number of total remaining packets is scheduled. If none of the top-subtrees are eligible, the sink does not receive any packet during that time slot. Inside each top-subtree, nodes are scheduled according to the rules in lines 5–12. A subtree

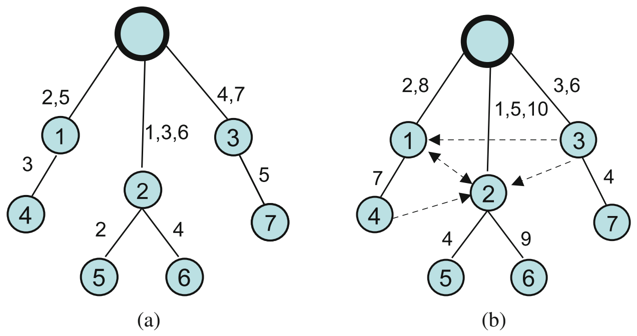


Fig. 14.6 Raw-data convergecast using algorithm LOCAL-TIMESLOTASSIGNMENT: (**a**) Schedule length 7 when secondary conflicts are eliminated. (**b**) Schedule length 10 when secondary conflicts are present

is defined to be *active* if there are still packets left in it (excluding its root) to be relayed. If a node's buffer is empty and the subtree rooted at this node is active, one of its children is scheduled at random whose buffer is not empty. The algorithm guarantees that in an active subtree there will always be at least one child whose buffer is not empty, and so whenever a node empties its buffer, it will receive a packet in the next time slot, thus emptying buffers from the bottom of the subtree to the top.

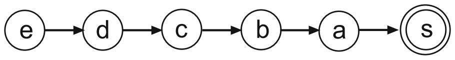
Figure. 14.6a shows an illustration of the working of the algorithm. In slot 1, since the eligible top-subtree containing the largest number of remaining packets is $\{2, 5, 6\}$, link $(2, s)$ is scheduled and the sink receives a packet from node 2. In slot 2, the eligible top-subtrees are $\{1, 4\}$ and $\{3, 7\}$, both of which have two remaining packets. We choose one of them at random, say $\{1, 4\}$, and schedule the link $(1, s)$. Also, in the same time slot since node 2's buffer is empty, it chooses one of its children at random, say node 5, and schedule the link $(5, 2)$. In slot 3, the eligible top-subtrees are $\{2, 5, 6\}$ and $\{3, 7\}$, both of which have two remaining packets. We choose the first one at random and schedule the link $(2, s)$, and so the sink receives a packet from node 5 (relayed by node 2). We also schedule the link $(4, 1)$ in slot 3 because node 1's buffer is empty at this point. This process continues until all the packets are delivered to the sink, yielding an assignment that requires seven time slots. Note that, in this example, $2n_k - 1 = 5$, and so $\max(2n_k - 1, N) = 7$. In Fig. 14.6b, an assignment is shown when all the interfering links are present, yielding a schedule length of 10.

A similar result of $\max(2n_k - 1, N)$ is obtained by Song et al. [66] where they also extended it to the case when the nodes have different number of packets to send. Assuming node i generates d_i packets, their proposed algorithm takes $\max\left(2\sum_{i\in \mathrm{TS}(r_k)} d_i - d_{r_k} + \sum_{i=2}^{k}(d_{r_i} - 1), N'\right)$ time slots, where $r_k, \ldots, r_1$ are the roots of the top-subtrees sorted in descending order of the total number of packets generated in it, k is the total number of top-subtrees, and N' is the total number of packets in the whole network.

A similar result utilizing multiple channels that minimize the schedule length for raw-data convergecast in WirelessHART networks [65] is obtained by Zhang et al. [61]. One significant difference between WirelessHART networks and sensor networks is that the former performs channel hopping on a per-packet basis, while most existing TDMA convergecast schemes do not support this feature. Thus, parallel transmissions scheduled in the same time slot must use different channels, whereas most of the existing TDMA-based multi-channel protocols first statically assign channels to eliminate potential interference and then perform time slot scheduling. Like in [37] and [66], they also consider buffer requirements at each node and show that when nodes can store at most one packet, the minimum schedule length for line topologies is $2N - 1$ using at most $\lceil N/2 \rceil$ channels (see Fig. 14.7a and b). However, when the nodes can buffer multiple packets, the optimal convergecast time remains the same while the number of channels required can be reduced to $\lceil N - \sqrt{N(N-1)/2} \rceil$ (see Fig. 14.7c). The basic idea of their proposed approach, which is similar to [37] and [66], is to schedule as many transmissions as possible in each time slot in order to maximize the use of available channels and to make sure that a node which does not have a packet at the beginning of a time slot receives one packet at the beginning of the next time slot.

Zhang et al. also study the effects of packet copying between the microcontroller and the radio transceiver [75] and propose a novel method that separates packet copying from packet transmission in order to improve the schedule length for WirelessHART networks. They show that packet copying could create a bottleneck in the critical path of packet forwarding and presented in [54] a scheme called *conditional immediate transmission* (CIT) which nearly tenfolds multi-hop 802.15.4 throughput.

The basic idea of CIT stems from the following observation. After receiving a packet, the microcontroller triggers an interrupt and notifies a process to fetch the incoming data over the serial peripherals interface (SPI) bus. Before transmitting a packet, the microcontroller first copies the packet into the radio's transmit buffer over the SPI bus and then sends a separate command to the radio to start transmission (see Fig. 14.8a). CIT removes this packet copying off the critical path by copying packet $n - 1$ into the transmit buffer before packet n arrives. When packet



(a)

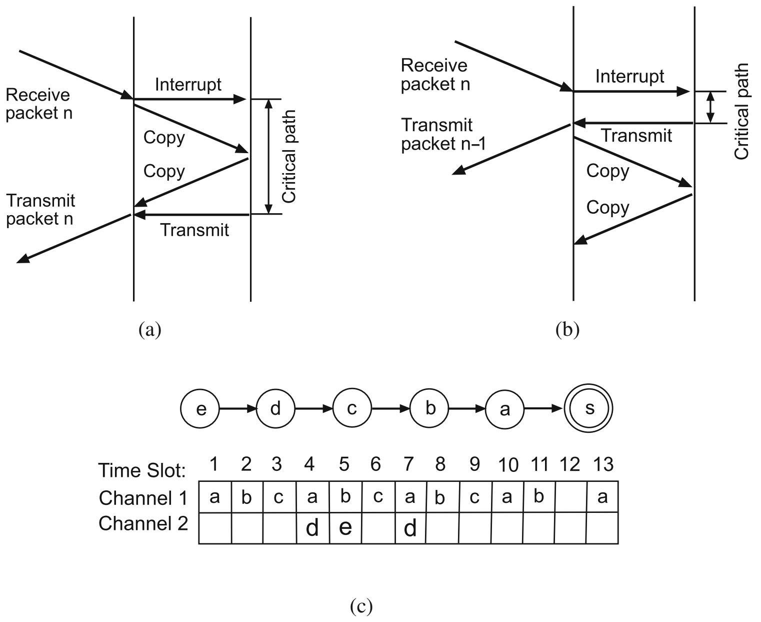
Time Slot:	1	2	3	4	5	6	7	8	9
Channel 1	a	b	a	b	a	b	a	b	a
Channel 2			c	d	c	d	c		
Channel 3					e				

(b)

Time Slot:	1	2	3	4	5	6	7	8	9
Channel 1	a	b	a	b	a	b	a	b	a
Channel 2	e	d	c	d	c		c		

(c)

Fig. 14.7 (**a**) A line network with five source nodes. (**b**) Schedule length is 9 using three channels when nodes can buffer at most one packet. (**c**) Schedule length is 9 using two channels when nodes can buffer multiple packets



Time Slot:	1	2	3	4	5	6	7	8	9	10	11	12	13
Channel 1	a	b	c	a	b	c	a	b	c	a	b		a
Channel 2				d	e		d						

Fig. 14.8 (**a**) Packet copying is on the critical path. (**b**) Packet copying is removed from the critical path. (**c**) Optimal CIT-based convergecast schedule for a line network of five source nodes

n arrives, packet $n - 1$ can be immediately forwarded without any copying (see Fig. 14.8b), and so the channel can be released immediately after transmitting/receiving and can be allocated to another node in the next time slot, thereby improving channel utilization. The authors show that although the number of time slots needed increases, the length of each slot is significantly reduced, enabling convergecast with higher throughput.

For line networks with N source nodes each with one packet to send, they show that the lower bound to complete a CIT-based convergecast in the presence of at most $\lceil N/3 \rceil$ channels is $3N - 2$. They also propose an algorithm that achieves this lower bound, requiring no node to buffer more than one packet at any time slot. The basic idea, as illustrated with a five-node network in Fig. 14.8c, is to schedule transmission for the immediate child of the sink at every $t = 3m + 1$ time slots, for $m = 0, \ldots, N - 1$ and for every other node, which still has packets to forward from its subtree, at slot t if its parent was scheduled at slot $t - 1$. This idea is extended to tree networks where the root of a top-subtree with the maximum number of remaining packets is scheduled every three time slots. Within each top-subtree a node is scheduled at slot t if its parent was scheduled at slot $t - 1$, and the subtree rooted at this node has the maximum number of remaining packets among all the subtrees of the top-subtree. The algorithm requires $\max\{3n_k + \beta, N\}$ time slots, which is optimal to complete CIT-based convergecast on a tree, requiring number of channels at most equal to the depth of the tree. Here $\beta = -1$ if $n_k = n_{k-1}$, and

$\beta = -2$ otherwise, and n_k, as in [37] and [66], is the maximum number of nodes in any subtree sorted in decreasing order of sizes $n_k \geq n_{k-1} \geq \cdots \geq n_1$.

14.3.1.2 Aggregated Data Convergecast

Unlike raw-data convergecast where the application requires every single packet generated by the nodes to be delivered to the sink, periodic data collection often requires delivery of only summarized information in the form of aggregated packets. In general, such aggregated convergecast requires less number of time slots than raw-data convergecast because of the reduced volume of traffic en route to the sink. Under this setting, it is assumed that every node generates a single packet at the beginning of every frame and perfect data aggregation is possible, i.e., each node is capable of aggregating all the packets received from its children as well as that generated by itself into a single packet before transmitting to its parent. This means that the size of aggregated data is constant and does not depend on the actual raw sensor readings. Typical examples of such aggregation functions are MIN, MAX, MEDIAN, COUNT, AVERAGE, etc., which are known as *algebraic* and *distributive* functions [49].

Since the goal is to minimize the schedule length, each parent node ideally should wait to receive all data from its children and then aggregate those with its own data before transmitting. Thus, in aggregated convergecast, a node transmits only once per frame and it maintains an intrinsic order of transmission with respect to its children. When the routing tree is not specified as part of the application requirements, the algorithms in this category also construct the routing tree suitable for aggregation and then perform scheduling. Two of the studies that we discuss below also consider multiple frequencies to eliminate interference. In the following, we discuss several of these algorithms that address the aggregated convergecast problem and its variants.

One of the early works is by Chen et al. [8], where the problem is slightly generalized by considering only a subset $S \subseteq V$ of nodes generating data instead of all the nodes. Assuming uniform transmission range and a unit disk graph (UDG) model, they formulate it as a *minimum data aggregation time* (MDAT) problem where the goal is to find a collision-free schedule that routes data from the subset of nodes to the sink in the minimum possible time. They prove that MDAT is NP-complete, even when restricted to UDGs, by reducing it from the restricted planar 3-SAT problem and design a centralized $(\Delta - 1)$-approximation algorithm, where $\Delta + 1$ is the maximum number of nodes within the transmission range of any node. Their proposed approach does not assume that the routing tree is known a priori; instead, the algorithm finds the data aggregation tree after the schedule is made. If the height of the routing tree is h, then a trivial lower bound on the schedule length is $\max\{h, \log_2 |S|\}$.

The basic idea of the algorithm, called the *shortest data aggregation* (SDA), is to incrementally construct smaller and smaller shortest path trees (SPT) rooted at the sink that span nodes possessing all the data, i.e., in the current iteration, the SPT rooted at the sink spans a set of nodes that possess all data aggregated from S till the previous iteration. The current iteration produces a collision-free schedule that comprises a set of simultaneously transmitting senders, which are selected from the leaves of the SPT based on the number of non-leaf neighbors in the graph, and a set of corresponding receivers.

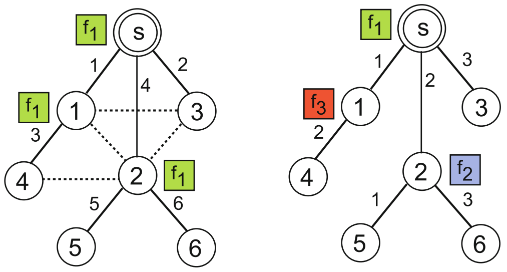
Malhotra et al. [3] consider the joint routing and scheduling problem for aggregated convergecast with the goal to construct an optimal routing tree that will help minimizing the schedule length. The basic idea of the tree construction algorithm is to create a shortest path tree and balance the number of children per node so that more parallel transmissions can take place without any single node causing bottleneck. The authors show that for a given routing tree, a lower bound on the schedule length is $\max_{i \in V}\{\xi_i + h_i\}$, where ξ_i and h_i are the number of children and hop distance from the sink, respectively, for node i. To balance the number of children per node, an optimal *semi-matching* formulation on bipartite graphs, originally due to Harvey et al. [35], is used where the goal is to assign nodes from level $h+1$ to the parents at level h such that every parent has an equal number of children. Once the balanced tree is constructed, a ranking-based heuristic is used for scheduling, where the idea is to rank all eligible nodes in decreasing order of their weights which are taken as the number of *non-leaf neighbors*. A higher weight gives a higher relative priority to a node to be scheduled in the current slot over other eligible nodes.

A variation of the aggregated convergecast problem where nodes can adjust their transmission ranges is studied by Shang et al. [63]. They propose an approximation algorithm that has a constant factor guarantee on the optimal schedule length for unit disk graphs. It first constructs a BFS tree rooted at the sink and then constructs a *maximal independent set* using the greedy First-Fit algorithm by choosing nodes in order of their increasing hop distances from the sink in the BFS tree. Note that this results in a *dominating set* which contains the sink but is not a connected set. Then, a minimal number of connector nodes are added to construct a *connected dominating set*, V_{CDS}, and the transmission ranges of all the nodes in this set are set to 1. Next, the scheduling phase runs in two stages. In the first stage, nodes in $V \setminus V_{\mathrm{CDS}}$ are scheduled first so that all their data reach the nodes in V_{CDS}. In the second stage, data is sent from the nodes in V_{CDS} to the sink. It is shown that the first stage takes $15 \log_2 |V \setminus V_{\mathrm{CDS}}|$ time slots, whereas the second stage takes $16d(T_{\mathrm{BFS}}) - 12$ time slots, where $d(T_{\mathrm{BFS}})$ is the depth of the BFS tree. Combining these two, it is shown that the schedule length is at most 31 times the optimal.

Zhang et al. extend their work which was focused on minimal time convergecast scheduling for raw-data convergecast [26] to aggregated convergecast scheduling in [77]. When the size of data is much smaller than the size of the data frame, nodes aggregate the received packets instead of sending single packets. They use the same scheduling algorithm proposed in [26, 27]. They show that using their scheduling algorithm with packet aggregation requires at most $N+2$ time slots in a linear network with N nodes. According to the algorithm, the sink receives the first packet in the first time slot. Then in every three time slots it receives an aggregated packet which contains data from three original unaggregated packets, due to two-hop scheduling to prevent interference. Hence, the total number of required time slots is $1 + 3\left\lceil \frac{N-1}{3} \right\rceil \leq N + 2$. For multi-line networks, the algorithm achieves a schedule length of $\max\left(n_k + 3\left\lceil \frac{N+2k}{3} \right\rceil\right)$, where k is the number of branches and n_k is the maximum number of nodes in a branch. Finally, for tree networks aggregation-enabled convergecast requires $\max\left(\hat{n}, \left\lceil \frac{N+2L+2(L-k)}{3} \right\rceil\right)$, where L is the number of leaf nodes, k is the number of one-hop subtrees, $\hat{n} = \max_i(n_i + 4l_i - 2)$, n_i is the number of nodes, and l_i is the number of leaf nodes in the ith one-hop subtree.

A variant of the aggregated convergecast problem where a parent node need not wait to receive all data from its children within a single frame before transmitting is investigated by Incel et al. [38] and Ghosh et al. [30]. This is particularly applicable for continuous and periodic monitoring applications that sustain over long durations of time. As explained in the following, the transmission ordering constraint between a parent node and its children within a single frame disappears once a *pipeline* is established, after which the sink starts receiving aggregated data from all the nodes in the network once every frame. In [38], the problem is studied through extensive simulations and experiments, whereas its theoretical aspects are discussed in [30]. In both works, multiple frequency channels are considered as means to eliminate interference. In the following, we first explain the notion of schedule length and pipelining in this variant of aggregated convergecast.

In Fig. 14.9, we show a network of six source nodes where the solid lines represent tree edges and the dotted lines represent interfering links. The numbers beside the links represent the time slots at which the links are scheduled to transmit. The



	Frame 1						Frame 2					
	S1	S2	S3	S4	S5	S6	S1	S2	S3	S4	S5	S6
s	1	2	3	-	-	-	{ 1,4 }	{ 2,5,6 }	3	-	-	-
1	-	-	-	4	-	-	-	-	-	4	-	-
2	-	-	-	-	5	6	-	-	-	-	5	6

(c)

Fig. 14.9 Aggregated convergecast: (**a**) Schedule length of 6 for single frequency. (**b**) Schedule length of 3 when multiple frequencies are used to eliminate interference. (**c**) Node IDs from which aggregated data is received by their corresponding parents in each time slot over different frames

frequencies assigned to the receivers of the tree are shown in boxes. The entries in the table list the nodes from which packets are received by their corresponding receivers in each time slot for Fig. 14.9a. We note that at the end of frame 1, the sink does not have packets from nodes 5 and 6; however, as the same schedule is repeated, it receives aggregated packets from nodes 2, 5, and 6 in slot 2 of the next frame. Similarly, the sink also receives aggregated packets from nodes 1 and 4 starting from slot 1 of frame 2. The entries $\{1, 4\}$ and $\{2, 5, 6\}$ in the table represent single packets comprising aggregated data from nodes 1 and 4 and from nodes 2, 5, and 6, respectively. Thus, a pipeline is established from frame 2, and the sink continues to receive aggregated packets from *all* the nodes once every six time slots. Thus, the minimum schedule length is 6. However, if all nodes are assigned different frequencies, as shown in Fig. 14.9b, then the minimum schedule length turns out to be 3.

The authors in [38] explore a number of different techniques that provide a hierarchy of successive improvements, the simplest among which is an interference-aware, minimum-length TDMA scheduling that enables spatial reuse. To achieve further improvement, they combine transmission power control with scheduling and use multiple frequency channels to enable more concurrent transmissions. The multiple frequencies are assumed to be orthogonal, and a *receiver-based channel assignment* (RBCA) scheme is proposed where the receivers (i.e., parents) in the tree are statically assigned different frequencies to eliminate interference. It is shown through extensive simulations that once multiple frequencies are used along with spatial-reuse TDMA, the data collection rate often no longer remains limited by interference, but by the topology of the network. Thus, in the final step, *degree-constrained trees* are constructed that further enhances the data collection rate.

Ghosh et al. in [30] prove that minimizing the schedule length under multiple frequencies is NP-hard on general graphs and propose approximation algorithms with worst-case provable performance guarantees for geometric networks. In particular, they design a constant factor approximation algorithm for *unit disk graphs* where every node has a uniform transmission range and a $O(\Delta(T) \log n)$ approximation for general disk graphs where nodes have different transmission ranges, where $\Delta(T)$ is the maximum node degree in the routing tree. They also show that a constant factor approximation is still achievable when the routing topology is not known a priori so long as the maximum node degree in the tree is bounded by a constant. Among other theoretical results, Yu et al.[72] proposed a greedy distributed aggregation scheme that takes at most $24D + 6\Delta + 16$ slots, where D is the network diameter and Δ is the maximum node degree.

14.3.2 Algorithms on Minimizing Latency

Minimizing the schedule length to complete convergecast certainly contributes to minimizing latency in data collection; however, in certain cases, it does not guarantee minimizing the average latency for individual packets. For instance, in aggregated data collection, where each sensor node is scheduled once per frame, and instead of relaying individual packets, they aggregate packets before forwarding toward the sink node, the minimal schedule length is equal to the maximum degree of the routing tree, as shown in [38] and discussed in Sect. 14.3.1. If causality is important, such that a node needs to wait for data from its children before being scheduled (i.e., when a pipeline, as discussed in Sect. 14.3.1, cannot be established), data collection cannot be completed in a period equal to this minimal schedule length. In this section, we survey algorithms that identify minimizing latency as their primary objective.

In [13], Cui et al. focus on minimizing the latency and analyzing the energy latency trade-off for data collection in WSNs where each node generates the same number of packets within each frame of length T. Data is transmitted to a relay node which forwards it toward the sink node; relay nodes are assumed to be not generating data. The authors first present sufficient conditions on link scheduling in order to achieve the minimum worst-case latency T and then present a link scheduling algorithm satisfying these conditions. They propose and prove that it is sufficient for every node to schedule its outgoing links *after* its incoming links in order to achieve the minimum possible latency T. The proposed algorithm classifies the links into levels according to their distance in number of hops from the sink, and the schedule is constructed in reverse order of hop distance, as illustrated in Fig. 14.10. A similar study is presented in [16], where it is shown that minimum-length scheduling does not automatically guarantee minimum latency, and a heuristic is proposed to minimize latency by scheduling the incoming links before the outgoing links.

In [56], Pan et al. propose algorithms for quick convergecast in ZigBee tree-based WSNs. The objective is to enable quick convergecast operations with minimum latency and complying with the ZigBee standard. Different from the studies presented in [12, 27], which minimize latency by minimizing the schedule length and assigning slots to the senders, this study considers receiver-based scheduling. This is due to the fixed wake-up/sleep scheduling specified in the ZigBee stack: in each cycle, nodes wake up twice, first to receive packets from their children and

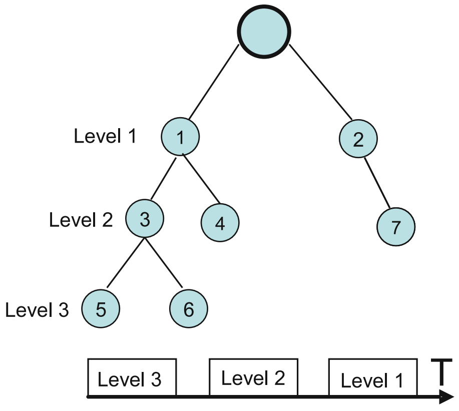


Fig. 14.10 Scheduling to minimize latency

second to transmit to their parents in a ZigBee beacon-enabled tree network. The authors first define a minimum latency beacon scheduling problem for quick convergecast in ZigBee networks and prove it to be NP-complete. Then they propose an algorithm which gives optimal performance for line topologies and within 1.5 times the optimal for ring topologies. The algorithm is also extended for tree-based schemes as a heuristic. A centralized tree-based algorithm traverses the nodes on a tree in a bottom-up manner, starting with the leaf nodes, similar to the previously discussed algorithm [13]. Nodes at the same depth of the tree are sorted according to the interference values (i.e., the number of links that cause interference on the link between the node and its parent) and starting with the most interfered node, and scheduling continues sequentially by assigning the first minimum available slot. Finally, a distributed version of the time slot assignment algorithm is proposed. The Performance of the proposed scheme is evaluated via extensive simulations, and the results are compared with random and greedy scheduling algorithms. Compared to the random and greedy schemes, the proposed heuristics can effectively achieve quicker convergecast. The performance of the heuristics decrease when the number of interference neighbors is high.

Revah et al. in [60] extends the work of Florens [22] by considering minimizing both average delivery time and completion time for convergecast. The authors argue that scheduling strategies that aim to minimize the completion time of a convergecast do not take into account the idle time of messages. For instance, it is unreasonable not to transmit a message toward the destination if it can be transmitted without any delay. Polynomial time solutions are presented for different network topologies: linear, two-branch, and star (or multi-branch) network. The sink node is assumed to have full information about the network topology and computes the schedules on demand. They show that in order to avoid possible delays, an algorithm should start immediate transmission of packets belonging to different groups, such that packets belonging to some group can be transmitted without being delayed by packets from other groups. Although the presented algorithms are centralized, they provide lower bounds for the problem. The nodes are assumed to be equipped with directional radios with two separate control channels for upstream and downstream communication. The protocol interference model is used in the analysis. Evaluations of the algorithms over tree networks are missing in the paper.

Another variant of scheduling is considered by Lu et al. in [46] where joint scheduling and routing with minimum latency requirement is studied. They define the *minimum latency joint scheduling and routing* (MLSR) problem as follows: Given a graph $G = (V, E)$, number of slots K, and flows M, the goal is to find paths P and a slot assignment f such that it maximizes the number of flows with minimum average latency. The problem is solved using a graph coloring approach: First a delay graph is constructed and it is shown that the minimum-weight M node-disjoint paths in the delay graph can be mapped to a solution of MLSR. In the next step, their proposal iteratively finds M node-disjoint paths.

In [40, 43, 47], the performance of different sleep scheduling techniques to minimize latency is evaluated. Although the focus is on sleep scheduling where nodes stay in low-power or sleep modes for most of the time, periodically waking up to check for activity, the proposed algorithms define the transmission rights for nodes similar to TDMA scheduling. Fully synchronized pattern, shifted even–odd pattern, ladder pattern, two-ladder pattern, multi-parent pattern, and multi-clustering pattern are explored, and their latency behaviors are analyzed in terms of minimum, maximum, and average latency.

14.3.3 Algorithms with Other Objectives

Besides the well-studied objectives of minimizing the schedule length and latency, which are the main focus of this chapter, there also exist studies that focus on other criteria. In this section, we briefly survey some of these studies with different objectives, such as minimizing energy and transmission power, maximizing capacity, maximizing fairness, and meeting deadlines.

14.3.3.1 Algorithms on Minimizing Energy

Energy efficiency is the biggest challenge in designing long-living sensor networks. Since radio communication consumes a lot of energy, a common method is to operate the radio with duty cycling that periodically switches the radio between sleep and wake-up modes. TDMA-based protocols offer the advantage of permitting nodes to enter into sleep mode during inactive periods, thus achieving low duty cycles and conserving energy. Additionally, TDMA-based medium access efficiently eliminates collisions and prevents overhearing, which are the main sources of energy consumption in wireless communication. Therefore, all the TDMA-based protocols proposed for WSNs have the inherent objective of minimizing energy consumption. Transmission power control is one of the well-studied methods in minimizing energy consumption and alleviating interference in wireless networks. Excessive levels of interference can be eliminated if the signals are transmitted with just enough power instead of maximum power.

In [39], Kalpakis et al. consider the maximum lifetime data-gathering problem, with and without aggregation, and propose polynomial time algorithms for maximizing the network lifetime, which is defined as the time until the first sensor runs out of energy. They formulate the problem for the aggregation case as a network flow problem using an ILP and propose an iterative algorithm called, *maximum lifetime data aggregation* (MLDA), to find a maximum lifetime schedule. The case without aggregation, called the *maximum lifetime data routing* (MLDR) problem, is formulated as a maximum flow problem with energy budgets on the nodes and again solved using an ILP.

In [51], a TDMA scheduling scheme for many-to-one communication is studied. TDMA-based communication provides a common energy-saving advantage by allowing nodes to turn their radio off when not engaged in communication; however, too much state transitions between the active and the sleep modes can waste energy. Accordingly, the desired objectives in this paper are to minimize the total time for data collection as well as to minimize the energy consumed on switching between the active and sleep states. To solve this optimization problem, two population-based stochastic optimization techniques, *particle swarm optimization* and *genetic algorithm*, are hybridized. The former guarantees that there is no empty slot during scheduling, and the latter ensures a strong searching ability to find the optimal slot allocation. It is shown by simulations that the hybrid algorithm outperforms the particle swarm optimization algorithm and the coloring methods in terms of the energy efficiency and finding minimal schedule lengths.

In [18], ElBatt et al. study the problem of joint scheduling and power control. Although the ideas presented in this paper are not directly targeted for WSNs, the problem of joint power control and TDMA scheduling also arises in WSNs, and the solution presented in the paper has been used for minimizing the data collection time in [38]. The algorithm proposed in [18] is a cross-layer method for joint scheduling and power control to improve the throughput capacity. The goal is to find a TDMA schedule that can support as many transmissions as possible in every time slot. It has two phases: (i) scheduling and (ii) power control. The scheduling phase searches for a *valid transmission schedule* where no node is to transmit and receive simultaneously or to receive from multiple nodes simultaneously. The power control phase then iteratively searches for an *admissible schedule* with power levels chosen to satisfy all the interfering constraints in the given valid schedule. In each iteration, the scheduler adjusts the power levels depending on the current RSSI at the receiver and the SINR threshold according to the iterative rule: $P_{\text{new}} = \frac{\beta}{\text{SINR}} \cdot P_{\text{current}}$, which is the well-known power control algorithm by Foschini and Miljanic [25]. If the maximum number of iterations is reached and there are nodes which cannot meet the interfering constraints, the scheduling phase excludes the link with minimum SINR. The power control phase is then repeated until an admissible transmission scenario is found.

The problem of joint scheduling and transmission power control is studied by Moscibroda [52], which will be surveyed in the next section for constant and uniform traffic demands in WSNs. In this paper, it is shown that unbounded improvements in the asymptotic capacity of data collection can be achieved by employing nonlinear power assignment at nodes.

14.3.3.2 Algorithms on Maximizing Capacity

Although maximizing the throughput capacity is not considered to be one of the prioritized objectives in traditional low-rate data collection applications, it may become an important concern in dense deployment or during certain periods when large bursts of packets are generated, for instance, due to a change in the monitored conditions. Moreover, with the adoption of WSNs in different application areas, such as industrial monitoring, health care, and surveillance where large amounts of data need to be collected and sometimes streamed, maximizing the capacity with limited resources has become a popular topic [17]. In this section, we describe two data collection studies that focus on this objective.

In [17], Duarte et al. present ideas which, although not based on TDMA scheduling, are very closely related to the problem of many-to-one data gathering. The trivial upper bound per node is presented as W/N (W is the transmission capacity and N is the number of nodes), which can be achieved when the sink is 100% busy in receiving. They show circumstances under which this bound is achievable, such as when all the sources can directly transmit to the sink node. On the other hand, if each source cannot directly reach the sink node and the communication takes place in multiple hops, it may or may not be possible to achieve the upper bound depending on the transmission and interference ranges. The bounds presented in the capacity domain can be translated into bounds on schedule length in TDMA. The trivial upper bound for the minimal schedule length is N, which similarly can be achieved when all the sources can directly transmit to the sink node.

The worst-case capacity of wireless sensor networks is studied by Moscibroda in [52], where it is theoretically shown that nonlinear power control mechanisms (without discrete power levels) can significantly help in minimizing the scheduling complexity and also in improving the capacity of WSNs. In this work, the aggregated data capacity and the notion of worst-case capacity, which concerns the question of how much information can each node transmit to the sink, regardless of the network's topology, are investigated for typical *worst-case* structures, such as chains. They prove that the achievable rate in the protocol model is $\theta(1/N)$, whereas it is $\Omega(1/\log_2 N)$ in the physical SINR model using transmission power control techniques (N is the number of nodes). For instance, in Fig. 14.11, two time slots are required to schedule both transmissions in the protocol model, but in the physical model both transmissions can be scheduled simultaneously by transmitting at appropriate power levels. Hence, the most important conclusion of this study is that there is an exponential gap between these two interference models in terms of achievable data rates. However, the assumption that the transceivers can set their power level to any value without considering discrete power levels and limits on the transmission power is somewhat limiting.

Motivated by Moscibroda's findings on using SINR-based interference model for computing capacity, Chafekar et al. in [6] develop polynomial time algorithms to provably approximate the total throughput. They define a throughput maximization problem with SINR constraints: Given a set of nodes V, a set of source–destination pairs, and a transmission power level at each edge, the problem is to (a) choose routes for the pairs, (b) choose flow rates on the routes, and (c) schedule packets at each time slot respecting SINR constraints for all parallel transmissions, such that the total throughput capacity is maximized. They develop a linear programming formulation to approximate the maximum throughput rate vector for SINR constraints. For the case of uniform power levels, they develop a polynomial time

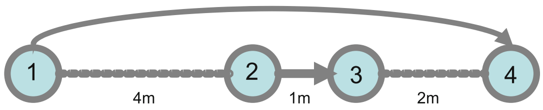


Fig. 14.11 Transmissions from node 1 to node 2 and from node 3 to node 4 can simultaneously take place in the physical interference model, whereas two time slots are needed in the protocol model

approximation algorithm that finds a feasible rate vector with a total throughput of at least $\Omega(r_{\text{opt}}/\log \Delta)$, where r_{opt} is the maximum possible throughput for an instance and Δ is defined as $\max_{u,v \in V} d(u,v)/\min_{u',v' \in V, u' \neq v'} d(u',v')$, where u, u', v, v' represent edges. When non-uniform power levels are used, they show that $O(\log \Delta \log \Gamma)$ approximation can be achieved, where Γ is the ratio between maximum and minimum power levels used. Similarly, in [21], approximation algorithms for link scheduling with SINR models are proposed.

14.3.3.3 Algorithms on Maximizing Fairness

In WSNs, fairness issue may arise as a problem under high data rates, for instance, when the data rates are comparable to available channel bandwidths. In this case, traditional randomized access schemes face the problem of unfair data delivery.

In [67], Sridharan et al. study max–min fair collision-free scheduling in WSNs by developing a linear programming formulation. They propose a distributed max–min fair scheduling mechanism suited for continuous traffic on a data-gathering tree and incorporate it with a time slot-based bandwidth allocation scheme to guarantee collision-free traffic. The algorithm for max-min fair resource allocation works in rounds, and at each round source nodes that are not constrained increase their generated data rate by a small incremental value ε. Nodes become constrained when the total bandwidth usage (the combination of incoming data, generated data, and interfering data traffic) at those nodes is within ε of the total bandwidth. Also, all nodes that are in the subtree below a constrained node, and all nodes whose output traffic interferes at a constrained node also become constrained. The algorithm terminates when all nodes on the routing tree become constrained and the rate available to all sources is the allocated rate. The scheme is incorporated with a TDMA-based scheduling algorithm with the goal to provide enough number of time slots for the data originating at each source at each frame. First, the root node allocates the required number of time slots to each of its children. The time slot allocation algorithm then runs in an iterative manner in a BFS order. At each iteration, only one node allocates time slots to its children. Simulations show that it outperforms an overhearing avoidance MAC (similar to S-MAC) and pruned 802.11 (without ACK's) in terms of energy consumption, fairness, and delay.

Another TDMA-based scheduling algorithm, called AI-LMAC, focusing on fairness was introduced by Chatterjea et al. in [7], which is an extension to the schedule-based MAC protocol, LMAC [36]. In the original LMAC protocol, each node on a data collection tree is allocated one time slot at each frame, whereas in AI-LMAC, nodes are assigned multiple time slots according to the traffic flowing over them. This ensures fairness such that the bandwidth allocated to a node corresponds to the traffic it is expected to encounter. Similar to the scheduling scheme proposed in [67], time slot allocation is based on a parent–child relationship over a data-gathering tree. A parent advises a child, i.e., sends a message to every one of its children indicating the ideal number of slots that a particular node should take up under the current traffic conditions. The process starts at the root node and percolates down the branches of the tree toward the leaf nodes.

14.3.3.4 Algorithms on Meeting Deadlines

Collecting data from a WSN within a specific deadline is closely related with the objectives of minimizing the schedule length and latency. However, it puts an additional constraint associated with the maximum latency allowed per message. For instance, in safety and mission-critical applications where sensor nodes are deployed to detect events, such as oil/gas leak or structural damages, the actuators or controllers need to receive data from all the sensors within specific deadlines [10]. Failure to receive data within the deadlines even from a single sensor may lead to unpredictable and catastrophic failures.

Scheduling messages with deadlines in multi-hop, real-time WSNs is first studied by Li et al. in [44]. They focus on the problem of providing timeliness guarantees for multi-hop transmissions in a real-time robotic sensor application where each message has a specific deadline. They show that the problem of meeting message deadlines is NP-hard and propose heuristics with deadline constraints. A central scheduler schedules messages based on their per-hop timeliness constraints and associated routes, transmission ranges, and the location of the nodes. First, the scheduler divides the transmission requests into disjoint sets such that transmissions within each set do not interfere with one another. While constructing these sets, the scheduler considers an order according to the latest transmission time (LST). The transmission with the minimum LST is chosen at each iteration and is assigned to a set where it does not interfere with other parallel transmissions without missing its deadline and without causing other transmissions in the same set to miss their deadlines. Performance of the proposed algorithm is compared with a method based on CSMA-CA where nodes make local scheduling decisions independent of others. Given a set of messages that are queued up at a node, the node schedules the message with the smallest LST. The proposed algorithm outperforms CSMA-CA-based scheduling in terms of the deadline miss ratio, especially when the utilization is high and/or the probability of collisions is high.

14.3.4 Algorithms with Joint Objectives

Besides the TDMA scheduling algorithms focusing on a particular objective for data collection in WSNs, another approach is to consider multiple objectives and address their trade-offs. One of the most studied joint objectives is minimizing latency together with minimizing energy consumption [74].

In [51], Mao et al. propose a TDMA scheduling scheme with the objective of minimizing the total time for completing a convergecast and minimizing the energy consumed on switching the transceiver between the active and the sleep states. The details of this algorithm is presented in Sect. 14.3.3.

Similar to [51], Wang et al. in [71] also focus on packet delay and the energy consumed on node state transitions. They propose a hierarchical solution to solve the multi-objective TDMA scheduling problem. Particle swarm optimization is exploited due to its strong search ability in combinatorial optimization and to reach multi-objective optimality. However, since there exists a conflict between delay and energy consumption in TDMA scheduling, the objectives cannot be optimized in parallel. For instance, after collecting data from its child nodes, the sensor node should wait to transmit packets to its own parent instead of switching off. As a result, delay performance is sacrificed. To solve this mutually conflicting multi-objective optimization problem, the concept of Pareto optimality was used in the evaluation system. Minimizing delay and energy consumption is presented as a case study in this paper for the evaluation of the proposed multi-objective optimization algorithm.

In [48], Macedo et al. propose a TDMA-based MAC protocol for latency–energy minimization and interference avoidance for alarm-driven, event-based WSN applications, such as surveillance of sensitive areas. The authors emphasize the trade-off between end-to-end delay and energy efficiency. As in the other protocols discussed in Sect. 14.3.2, a cascading time slot assignment is used to minimize the end-to-end transmission delay. With this slot assignment, the algorithm achieves very low duty cycles, since each node should only listen during the slots assigned to its children at the beginning of each frame and switch to sleep mode afterward. Allocation of slots is performed by parent nodes in a localized manner without requiring a central scheduler. The algorithm to assign the slots is based on Request To Assign and Clear to Assign messages exchanged by the parents and their children that are similar to RTS/CTS messages in CSMA-CA protocols. Instead of using a simple hop-based interference model, the algorithm checks the link quality directly experienced by the nodes.

In [68], Trigoni et al. propose wave scheduling and routing in sensor networks for energy efficiency in WSNs by considering data dissemination strategies that avoid collisions and message retransmissions at the cost of higher message latency. They define the energy minimization problem with the goal to determine a data dissemination scheme that minimizes the energy consumed in delivering all messages within a bounded delay. First, they prove that the problem is NP-hard. For the analysis, unit disk graphs are used such that two nodes are connected by an edge if and only if the Euclidean distance between them is at most 1. Moreover, to simplify the problem, they use a partitioning scheme that allows to schedule communication tasks at the cell level, rather than at the node level. They partition the network into square cells, where the length of each cell is set so that a node anywhere in a cell can typically communicate directly with nodes in adjacent cells. In their proposal, the scheduler schedules concurrently activate cells that are sufficiently spaced apart so that the message transmissions within these cells do not interfere with one another. In the wave scheduling algorithm, directed edges of the rectilinear grid that connect two adjacent cells are activated periodically in a sequential manner by avoiding interference at the MAC layer and allowing nodes to turn off their radios whenever they do not need to communicate. The algorithm jointly works with a routing protocol to reduce interference, and extensive simulations show that the proposed scheduling method results in significant energy savings at the expense of increased message latency.

Similar to [68], Oswald et al. present tight bounds for delay-sensitive aggregation in [55] by studying the trade-off between latency and energy cost. In order to reduce the communication cost, nodes aggregate messages while forwarding toward the sink node. In other words, nodes wait to receive/generate more packets before relaying so as to decrease the communication cost at the expense of late delivery at the sink node. The authors present competitive ratios (a metric, in which the performance of an on-line algorithm is compared to the performance of an optimal off-line algorithm) for a simple algorithm that aims to balance the total latency and energy cost in which nodes aggregate information about multiple events in one message until a forwarding condition is satisfied. The forwarding condition implies that a message should be forwarded to the parent node as soon as the current latency exceeds the transmission cost. They prove that any oblivious algorithm (i.e., given the packet arrival times all nodes react in the same way, independent of its distance to the sink node) achieves a competitive ratio of $O(\min(h, c))$, where h is the the height of the tree and c is the transmission cost per edge, for tree networks and $\Theta\left(\min\left(\sqrt{h}, c\right)\right)$ for chain networks.

In [42], Lee et al. study the scheduling problem with the objective of energy efficiency and reliability. In this work, besides collision-free scheduling, network flow optimization techniques and optimized tree topologies are used to achieve energy efficiency. Moreover, reliability is guaranteed by including many retransmission opportunities in the schedules. The protocol proposed gets the connectivity graph as the input and first assigns layers to the nodes to create a hierarchy in the network. The sink node is assigned to layer 0 and the nodes connected to the sink are assigned to layer 1. By using a BFS order, all nodes in the graph are assigned layers. After the layer assignment, a parent graph is created by removing the links among the nodes at the same layer. By using the parent graph, all the potential parent sets are created for the nodes. Based on the parent graph, optimal flow rates are calculated that minimize the energy consumption and accordingly usage percentages of links are calculated. Based on these percentages, the algorithm constructs a set of trees to be used in each data collection cycle. Finally, for each tree a collision-free schedule list is created by the sink node and the schedule is broadcast back to the nodes in the network.

14.3.5 Taxonomy

In this section, we provide a classification and a summary of the surveyed TDMA-based scheduling algorithms for data collection in WSNs. The classification is based on the design objectives, model constraints, and assumptions that are presented in Sect. 14.2. Table 14.1 summarizes the general aspects of the algorithms that are discussed in Sect. 14.3, ordered by the design objectives.

Most of the algorithms focus on the objective of minimizing the schedule length using an interference model that is a variant of the graph-based protocol model. Both aggregated and raw-data collection operations are investigated in the surveyed solutions. Most of the algorithms consider data collection operation over tree topologies, whereas some of them focus on specific topologies, such as chain, ring, or

Table 14.1 Taxonomy of TDMA-based data collection algorithms in WSNs

Reference	objective	Interference model	Data collection method	Topology	Hardware assumption	Implementation	Assignment scheme
[22–24]	Minimize schedule length	Protocol	Raw	Line, multi-line, tree	–	Centralized	Links
[20]	Minimize schedule length	Graph	Raw	Tree	–	Centralized	Links, nodes
[41]	Minimize schedule length	Interference range	Raw	Tree, disjoint strips	–	Centralized	Links
[12]	Minimize schedule length	2-hop	Raw	Line, tree	–	Centralized	Links
[27]	Minimize schedule length	2-hop	Raw	Line, multi-line, tree	–	Distributed	Links
[69]	Minimize schedule length	2-hop	Raw	Tree, general topology	–	Centralized	Links
[1]	Minimize schedule length	Graph	Raw	Tree	CDMA	Centralized	Links
[37]	Minimize schedule length	Protocol and Physical	Raw and Aggregated	Tree	Power control and multi-channels	Centralized	Links
[66]	Minimize schedule length	2-hop	Raw	Tree	Multi-channels	Centralized, distributed	Links
[61, 75]	Minimize schedule length	Measurements	Raw	Tree	Channel hopping	Distributed	Links
[8]	Minimize schedule length	Unit disk graph	Aggregated	Tree	–	Centralized	Links
[3]	Minimize schedule length	Graph	Aggregated	Tree	–	Centralized	Links
[63]	Minimize schedule length	–	Aggregated	Tree	Power control	Centralized	Links
[77]	Minimize schedule length	Interference range	Aggregated	Line, multi-line, tree	–	Distributed	Links

Table 14.1 (continued)

[30, 38]	Minimize schedule length	Protocol and physical	Aggregated	Tree	Power control and multi-channels	Centralized	Links
[13]	Minimize delay	–	Aggregated	Tree, general topology	–	Centralized	Links
[16]	Minimize delay	2-hop	Aggregated	Chain, general	–	Centralized	Links
[56]	Minimize delay	2-hop	Aggregated	Tree	–	Centralized, distributed	Links
[60]	Minimize delay	Protocol	–	Line, star, multi-line	Directional antennas	Centralized	Links
[46]	Minimize delay	Interference range	–	Tree	FDMA	Centralized, distributed	Links
[51]	Minimize total time and energy	2-hop	–	Tree	–	Centralized	Links
[71]	Minimize delay and energy	2-hop	–	General graph	–	Centralized	Links
[48]	Minimize latency and energy	Link quality — SINR	–	General	–	Distributed	Links
[68]	Trades off latency for energy	Cell based	–	General	–	Centralized	Links
[39]	Maximizing Lifetime	–	Raw and Aggregated	General graph	–	Centralized	Links
[55]	Minimize delay and energy	–	Aggregated	Chain, tree	–	Distributed	Links
[42]	Minimize delay and energy	Interference range	Aggregated	Dynamic trees	–	Centralized	Links
[17]	Maximize capacity	Protocol	Aggregated	Flat, clustered	–	Centralized, distributed	–
[52]	Maximize capacity, minimize power	Physical model	Aggregated	Line (worst case)	Power control	Centralized	Links
[6]	Maximize capacity, minimize power	Physical model	Aggregated	General	Power control	Centralized	Links
[67]	Maximize fairness	Interference range	Raw	Tree, general	–	Distributed	Links
[44]	Meeting message deadlines	Interference range	–	Star, tree	–	Centralized	Links

star. All of the algorithms require time synchronization to operate under a TDMA schedule, but none of them mention the granularity of the synchronization or the kind of synchronization schemes used. It is assumed that nodes are synchronized by a technique available in the literature [64]. Most of the algorithms work with simple transceivers available on the sensor nodes, while some only work with transceivers with a special capability, such as directional antennas. Additionally, some of the solutions explore the benefits of using transceivers that can adjust its transmission power level and/or operating frequency (these types of radios are commonly available on widely used sensor mote platforms [4, 62]), besides the use of simple radios. Most of the algorithms aim to find lower bounds or compute approximation algorithms for the studied NP-complete problems. Therefore, the algorithms provide centralized solutions which are usually computed by the sink node. Some of the algorithms present scheduling solutions that are coupled with additional methods, such as multiple frequencies, transmission power control, and CDMA to eliminate interference. The surveyed algorithms usually support all patterns of convergecast, such as one-shot or continuous data collection, while some are optimized for a specific one. Lastly, almost all the algorithms consider scheduling at link level rather than node level.

14.4 Future Research Directions/Open Problems

As we have seen, extensive research has been done with many different objectives in the field of TDMA scheduling for data gathering in WSNs. However, there still exist some open questions to be addressed, especially related to real implementation and evaluation of the proposals on testbeds or on real deployments. There also exist several theoretical questions that need to be addressed. In this section, we briefly summarize these open problems.

Most of the solutions are theoretical or simulation based (except [75]) offering only centralized solutions. Real implementation on sensor nodes where schedules are computed locally and are adaptive to network dynamics is necessary to enhance the operation of WSNs and to meet application requirements. For instance, we observe a trend in using WSNs to support more complex operations ranging from industrial control to health care, which require complex operations like detection of events in real time or responsive querying of the network by collecting streams of data in a timely manner. Thus, supporting QoS (quality of service) metrics such as delay and reliability become more important [31]. Therefore, distributed implementation and performance testing of the proposed algorithms on testbed or real deployments become essential. Additionally, real implementation and deployment will help in addressing the problems of intermittent connectivity and channel errors with unreliable links and handling asymmetric links. We should note that many of the existing algorithms provide lower bounds for the studied problems, and so they can be used as benchmarks for comparing the performance of the distributed solutions.

Although there exist some works that address multiple joint objectives, more detailed investigations to address the trade-offs between conflicting objectives will be beneficial. Most of the studies consider the trade-offs between energy efficiency and latency objectives. Different trade-offs can be identified between other objectives, such as minimizing latency and maximizing reliability or maximizing capacity and minimizing energy consumption. For instance, with the extension of WSNs in the visual domain where embedded cameras act as sensors, criteria such as reliability, QoS, and timeliness of the streamed data are becoming important. Solutions to address different objectives and trade-offs, for instance, consumed energy versus reconstructed image quality, should be explored within the perspective of data collection in WSNs.

Most of the surveyed algorithms consider fixed traffic patterns, i.e., every node generates a fixed number of packets in each data collection cycle. In a real scenario, some nodes may have a lot of packets that require more than one time slot per frame, while some others may not have any data to send in a time slot, thus wasting bandwidth. It will be interesting to explore the performance in such scenarios with random packet arrivals and combining the solutions of TDMA scheduling with rate allocation algorithms, especially in applications where high data rates are necessary.

In the surveyed papers, either raw-data collection or aggregated data collection are considered. Another possibility is to investigate different levels of aggregation, i.e., how much of the data received from the children is forwarded to the parent node. Investigating different levels of aggregation was proposed in [70] where the efficiency of different tree construction mechanisms was analyzed in terms of latency and energy metrics. This study can be extended for TDMA-based data collection algorithms in WSNs.

Some of the surveyed algorithms provide cross-layer solutions, where the schedules are computed together with methods such as transmission power control, optimal routing trees, and multi-frequency scheduling. It is indeed essential to address the problems from a cross-layer perspective to achieve the target functions and offer better performances. Along this line of research, Chafekar et al. in [5] extend the work by Moscibroda [53] in designing cross-layer protocols using the SINR model and proposed polynomial time algorithms with provable worst-case performance guarantee for the latency minimization problem. Their cross-layer approach chooses power level for all transceivers, routes for all connections, and constructs an end-to-end schedule such that SINR constraints are satisfied. A prominent research direction is to consider such cross-layer approaches from a theoretical point of view. More research can be done in this direction to combine the existing work with the solutions at different layers. In most studies, static topologies are assumed. Problems related to dynamic topologies, such as topological changes and addition of new nodes, are open. In addition, the time complexity of data gathering under various hypothesis [28], such as when some nodes have no packet to transmit, or when no buffering is allowed, remains open.

As was pointed by Moscibroda in [52], the type of interference model may heavily impact the achievable results. Use of realistic models for communication and interference is another direction that can be further investigated. Along this direction, Goussevskaia et al. [32] present the first NP-completeness proofs (by reducing from the partition problem) on two scheduling problems using the SINR interference model. The first problem consists in finding a minimum-length schedule for a given set of links. The second problem receives a weighted set of links as input and consists in finding a maximum-weight subset of links to be scheduled simultaneously in one shot. In [53], Moscibroda et al. study a generalized version of the SINR interference model and obtain theoretical upper bounds on the scheduling complexity of arbitrary topologies. They prove that if signals are transmitted with correctly assigned transmission power levels, the number of time slots required to successfully schedule all links in an arbitrary topology is proportional to the squared logarithm of the number of nodes times a previously defined static interference measure. More of such works that bridge the gap between static graph-based interference models and the physical models are needed.

14.5 Conclusions

In this chapter, we have surveyed TDMA-based scheduling algorithms for data collection in wireless sensor networks. We classified the algorithms according to their design objectives and constraints and provided a survey of existing algorithms with comparisons. In terms of the design objectives, most of the surveyed algorithms aim at (i) minimizing schedule length, (ii) minimizing latency, (iii) minimizing energy, and (iv) maximizing fairness, whereas some algorithms aim for joint objectives, such as maximizing capacity and minimizing energy or minimizing delay and energy. The surveyed algorithms also vary according to the design constraints and assumptions. For instance, some of the algorithms use simple models for communication and interference while some of them assume complex transceivers available on the nodes. As it was shown in some of the surveyed papers, the unrealistic models or assumptions may heavily impact the achievable results. Use of realistic models for communication, modification of those solutions that base their assumptions on unrealistic models, implementations on real sensor nodes and performance evaluations over real testbeds and deployments, and addressing the trade-offs between conflicting design objectives are identified as some of the important directions for future research.

References

1. V. Annamalai, S. Gupta, L. Schwiebert. On tree-based convergecasting in wireless sensor networks. In: *WCNC '03*, volume 3, pages 1942–1947, New Orleans, LA, USA, 2003.
2. K. Arisha, M. Youssef, M. Younis. Energy-aware tdma-based mac for sensor networks. System-level power optimization for wireless multimedia communication pages 21–40, 2002.
3. I.N. Baljeet Malhotra, M.A. Nascimento. Aggregation convergecast scheduling in wireless sensor networks. Technical report, University of Alberta, 2009.
4. CC2420: Single-chip 2.4 ghz ieee 802.15.4 compliant and zigbee(tm) ready rf transceiver. http://www.ti.com/lit/gpn/cc2420.

5. D. Chafekar, V.A. Kumar, M. Marathe, S. Parthasarathy, A. Srinivasan. Cross-layer latency minimization in wireless networks with SINR constraints. In: *MobiHoc '07*, ACM, New York, NY, pages 110–119, Montreal, Quebec, Canada, 2007.
6. D. Chafekar, V.S.A. Kumar, M.V. Marathe, 0002. S.Parthasarathy, A. Srinivasan. Approximation algorithms for computing capacity of wireless networks with SINR constraints. In: *INFOCOM*, pages 1166–1174, Phoenix, AZ, USA, 2008.
7. S. Chatterjea, L. van Hoesel, P. Havinga. Ai-lmac: An adaptive, information-centric and lightweight mac protocol for wireless sensor networks. In: *Issnip '04*, Melbourne, Australia, 2004.
8. X. Chen, X. Hu, J. Zhu. Minimum data aggregation time problem in wireless sensor networks. In: *MSN*, pages 133–142, Wuhan, China, 2005.
9. K. Chintalapudi, T. Fu, J. Paek, N. Kothari, S. Rangwala, J. Caffrey, R. Govindan, E. Johnson, S. Masri. Monitoring civil structures with a wireless sensor network. *IEEE Internet Computing*, 10(2): 26–34, 2006.
10. K.K. Chintalapudi, L. Venkatraman. On the design of mac protocols for low-latency hard real-time discrete control applications over 802.15.4 hardware. In: *IPSN '08*, pages 356–367, St. Louis, MO, USA, 2008.
11. I. Chlamtac, S. Kutten. Tree-based broadcasting in multihop radio networks. *IEEE Transactions on Computers* 36(10): 1209–1233 (1987)
12. H. Choi, J. Wang, E. Hughes. Scheduling for information gathering on sensor network. Wireless Networks (Online) (2007)
13. S. Cui, R. Madan, A. Goldsmith, S. Lall. Energy-delay tradeoffs for data collection in tdma-based sensor networks. In: *ICC '05*, volume 5, pages 3278–3284, Seoul, Korea, 2005.
14. M. Dalbro, E. Eikeland, A.J.i. Veld, S. Gjessing, T.S. Lande, H.K. Riis, O. Sør($\dot{a}$)sen. Wireless sensor networks for off-shore oil and gas installations. In: *SENSORCOMM '08*, pages 258–263, Cap Esterel, France, 2008.
15. I. Demirkol, C. Ersoy, F. Alagoz. Mac protocols for wireless sensor networks: A survey. *IEEE Communications Magazine* 44(4): 115–121, 2006.
16. P. Djukic, S. Valaee. Link scheduling for minimum delay in spatial re-use tdma. In: *Infocom '07*, pages 28–36, IEEE, Anchorage, Alaska, USA, 2007.
17. E. Duarte-Melo, M. Liu. Data-gathering wireless sensor networks: Organization and capacity. *Computer Networks* 43(4): 519–537, 2003.
18. T. ElBatt, A. Ephremides. Joint scheduling and power control for wireless ad-hoc networks. In: *Infocom '02*, volume 2, pages 976–984, 2002.
19. J. Elson, L. Girod, D. Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Operator Systems Review*, 36(SI): 147–163, 2002.
20. S. Ergen, P. Varaja. Tdma scheduling algorithms for sensor networks. Technical report, University of California, Berkeley, 2005.
21. S. Fan, L. Zhang, Y. Ren. Approximation algorithms for link scheduling with physical interference model in wireless multi-hop networks. CoRR abs/0910.5215, 2009.
22. C. Florens, M. Franceschetti, R. McEliece. Lower bounds on data collection time in sensory networks. *IEEE Journal on Selected Areas in Communications* 22(6): 1110–1120, 2004.
23. C. Florens, R. McEliece. Scheduling algorithms for wireless ad-hoc sensor networks. In: *Globecom '02*, pages 6–10, IEEE, Taipei, Taiwan, 2002.
24. C. Florens, R. McEliece. Packets distribution algorithms for sensor networks. In: *Infocom '03*, volume 2, pages 1063–1072, IEEE, San Francisco, CA, USA, 2003.
25. G. Foschini, Z. Miljanic. A simple distributed autonomous power control algorithm and its convergence. *IEEE Transactions on Vehicular Technology* 42(4): 641–646, 1993.
26. S. Gandham, Y. Zhang, Q. Huang. Distributed minimal time convergecast scheduling in wireless sensor networks. In: *ICDCS '06*, IEEE Computer Society, Washington, DC, page 50, 2006. DOI http://dx.doi.org/10.1109/ICDCS.2006.30
27. S. Gandham, Y. Zhang, Q. Huang. Distributed time-optimal scheduling for convergecast in wireless sensor networks. *Computer Networks* 52(3): 610–629, 2008.

28. L. Gargano. Time optimal gathering in sensor networks. In: *SIROCCO '07*, pages 7–10, 2007.
29. A. Ghosh. Estimating coverage holes and enhancing coverage in mixed sensor networks. In: *LCN '04*. IEEE Computer Society, Washington, DC, pages 68–76, 2004. DOI http://dx.doi.org/10.1109/LCN.2004.53
30. A. Ghosh, O.D. Incel, V.A. Kumar, B. Krishnamachari. Multi-channel scheduling algorithms for fast aggregated convergecast in sensor networks. In: *MASS '09*, pages 362–372, IEEE, Macau, China.
31. GINSENG: Performance control in wireless sensor networks. www.ict-ginseng.eu
32. O. Goussevskaia, Y.A. Oswald, R. Wattenhofer. Complexity in geometric SINR. In: *MobiHoc '07*, ACM, New York, NY, USA, pages 100–109, 2007. http://doi.acm.org/10.1145/1288107.1288122
33. J. Grönkvist, A. Hansson. Comparison between graph-based and interference-based stdma scheduling. In: *MobiHoc '01*, pages 255–258, ACM, Long Beach, CA, USA, 2001.
34. P. Gupta, P. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory* IT-46(2): 388–404, 2000.
35. N.J. Harvey, R.E. Ladner, L. Lovász, T. Tamir. Semi-matchings for bipartite graphs and load balancing. *Journal of Algorithms* 59(1): 53–78, 2006. http://dx.doi.org/10.1016/j.jalgor.2005.01.003
36. L. van Hoesel, P. Havinga. A lightweight medium access protocol (LMAC) for wireless sensor networks. In: *INSS' 04*. SICE (Society of Instrument and Control Engineers), Tokyo, Japan, 2004.
37. D.O. Incel, A. Ghosh, B. Krishnamachari, K. Chintalapudi. Fast data collection in tree-based wireless sensor networks. *IEEE Transactions on Mobile Computing* (*submitted*), 2009.
38. O.D. Incel, B. Krishnamachari. Enhancing the data collection rate of tree-based aggregation in wireless sensor networks. In: *SECON '08*, pages 569–577, IEEE, San Francisco, CA, USA, 2008.
39. K. Kalpakis, K. Dasgupta, P. Namjoshi. Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Computing Network* 42(6): 697–716, 2003.
40. A. Keshavarzian, H. Lee, L. Venkatraman. Wakeup scheduling in wireless sensor networks. In: *MobiHoc '06*, ACM, New York, NY, pages 322–333, 2006. http://doi.acm.org/10.1145/1132905.1132941
41. N. Lai, C.King, C. Lin. On maximizing the throughput of convergecast in wireless sensor networks. In: *GPC '08*, pages 396–408, Kunming, China, 2008.
42. H. Lee, A. Keshavarzian. Towards energy-optimal and reliable data collection via collision-free scheduling in wireless sensor networks. In: *INFOCOM*, pages 2029–2037, Phoenix, AZ, USA, 2008.
43. H. Lee, A. Keshavarzian, H.K. Aghajan. Multi-cluster multi-parent wake-up scheduling in delay-sensitive wireless sensor networks. In: *GLOBECOM*, pages 430–435, New Orleans, LA, USA, 2008.
44. H. Li, P. Shenoy, K. Ramamritham. Scheduling messages with deadlines in multi-hop real-time sensor networks. In: *RTAS 2005*, pages 415–425, San Francisco, CA, USA, 2005.
45. X.Y. Li, Y. Wang. Simple heuristics and ptass for intersection graphs in wireless ad hoc networks. In: *DIALM '02*, ACM, New York, NY, pages 62–71, 2002. http://doi.acm.org/10.1145/570810.570819
46. G. Lu, B. Krishnamachari. Minimum latency joint scheduling and routing in wireless sensor networks. *Ad Hoc Netw*. 5(6): 832–843, 2007. http://dx.doi.org/10.1016/j.adhoc.2007.03.002
47. G. Lu, N. Sadagopan, B. Krishnamachari, A. Goel. Delay efficient sleep scheduling in wireless sensor networks. In: *INFOCOM '05*, pages 2470–2481, Miami, FL, USA, 2005.
48. M. Macedo, A. Grilo, M. Nunes. Distributed latency-energy minimization and interference avoidance in tdma wireless sensor networks. *Computing Network* 53(5): 569–582, 2009. http://dx.doi.org/10.1016/j.comnet.2008.10.015
49. S. Madden, M. Franklin, J. Hellerstein, W. Hong. Tinydb: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems* 30(1): 122–173, 2005.

50. A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, J. Anderson. Wireless sensor networks for habitat monitoring. In: *WSNA '02*, pages 88–97, Atlanta, GA, USA, 2002.
51. J. Mao, Z. Wu, X. Wu. A tdma scheduling scheme for many-to-one communications in wireless sensor networks. *Computer Communications* 30(4): 863–872, 2007.
52. T. Moscibroda. The worst-case capacity of wireless sensor networks. In: *IPSN '07*, pages 1–10, Cambridge, MA, USA, 2007.
53. T. Moscibroda, R. Wattenhofer, A. Zollinger. Topology control meets SINR: The scheduling complexity of arbitrary topologies. In: *MobiHoc '06*, pages 310–321, 2006.
54. F. Osterlind, A. Dunkels. Approaching the maximum 802.15.4 multi-hop throughput. In: *HotEmNets 2008*, Charlottesville, VA, page 6, 2008. `http://eprints.sics.se/3426/01/osterlind08approaching.pdf`
55. Y.A. Oswald, S. Schmid, R. Wattenhofer. Tight bounds for delay-sensitive aggregation. In: *PODC '08*, ACM, New York, NY, pages 195–202, 2008. http://doi.acm.org/10.1145/1400751.1400778
56. M. Pan, Y. Tseng. Quick convergecast in zigbee beacon-enabled tree-based wireless sensor networks. *Computer Communications* 31(5): 999–1011, 2008.
57. C. Papadimitriou. The complexity of the capacitated tree problem. *Networks* 8(3): 217–230, 1978.
58. V. Rajendran, K. Obraczka, J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. In: *SenSys '03*, pages 181–192, 2003.
59. S. Ramanathan, E. Lloyd. Scheduling algorithms for multihop radio networks. *IEEE/ACM Transactions on Networking* 1(2): 166–177, 1993.
60. Y. Revah, M. Segal. Improved lower bounds for data-gathering time in sensor networks. In: ICNS '07, IEEE Computer Society, Washington, DC, page 76, 2007. http://dx.doi.org/10.1109/ICNS.2007.71
61. I. Rhee, A. Warrier, M. Aia, J. Min. Z-mac: A hybrid mac for wireless sensor networks. In: *SenSys '05*, pages 90–101, 2005.
62. Nordic Semi Conductors, nrf905 multiband transceiver. http://www.nordicsemi.com
63. W. Shang, P. Wan, X. Hu. Approximation algorithm for minimal convergecast time problem in wireless sensor networks. Wireless Networks, 2009. 10.1007/s11276-009-0207-9
64. F. Sivrikaya, B. Yener. Time synchronization in sensor networks: A survey. *IEEE Network* 18(4): 45–50, 2004. 10.1109/MNET.2004.1316761
65. J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon. Wirelesshart: Applying wireless technology in real-time industrial process control. In: *RTAS '08*, pages 377–386, St. Louis, MO, USA, 2008.
66. W.Z. Song, F. Yuan, R. LaHusen, B. Shirazi. Time-optimum packet scheduling for many-to-one routing in wireless sensor networks. International Journal Parallel Emergent Distributed Systems 22(5): 355–370, 2007. http://dx.doi.org/10.1080/17445760601111459
67. A. Sridharan, B. Krishnamachari. Max-min fair collision-free scheduling for wireless sensor networks. In: *IPCCC '04*, pages 585–590, Austin, TX, USA, 2004.
68. N. Trigoni, Y. Yao, A. Demers, J. Gehrke, R. Rajaraman. Wave scheduling and routing in sensor networks. *ACM Transactions on Sensor Networks* 3(1): 2, 2007.
69. H.W. Tsai, T.S. Chen. Minimal time and conflict-free schedule for convergecast in wireless sensor networks. In: *ICC '08*, pages 2808–2812, 2008.
70. S. Upadhyayula, S. Gupta. Spanning tree based algorithms for low latency and energy efficient data aggregation enhanced convergecast (dac) in wireless sensor networks. *Ad Hoc Networks* 5(5): 626–648, 2007.
71. T. Wang, Z. Wu, J. Mao. A new method for multi-objective tdma scheduling in wireless sensor networks using pareto-based pso and fuzzy comprehensive judgement. In: *HPCC '07*, Springer, Berlin, pages 144–155, 2007.
72. B. Yu, J. Li, Y. Li. Distributed data aggregation scheduling in wireless sensor networks. In: *Infocom '09*, Rio de Janeiro, Brazil, 2009.

73. L. Yu, N. Wang, X. Meng. Real-time forest fire detection with wireless sensor networks. In: *WiCom*, volume 2, pages 1214–1217, 2005.
74. Y. Yu, B. Krishnamachari, V.K. Prasanna. Energy-latency tradeoffs for data gathering in wireless sensor networks. In: *INFOCOM*, Hong Kong, China, 2004.
75. H. Zhang, F. Österlind, P. Soldati, T. Voigt, M. Johansson. Time-optimal convergecast with separated packet copying. Technical report, Royal Institute of Technology (KTH) (2009)
76. H. Zhang, P. Soldati, M. Johansson. Optimal link scheduling and channel assignment for convergecast in linear wirelessHART networks. In: *WiOPT '09*, Seoul, Korea, 2009.
77. Y. Zhang, S. Gandham, Q. Huang. Distributed minimal time convergecast scheduling for small or sparse data sources. In: *RTSS '07, IEEE Computer Society*, Washington, DC, pages 301–310, 2007. http://dx.doi.org/10.1109/RTSS.2007.19

Chapter 15
Position-Based Routing in Wireless Ad Hoc and Sensor Networks

Nathalie Mitton, Tahiry Razafindralambo, and David Simplot-Ryl

Abstract Geometric routing protocols are a memoryless and scalable approach which uses position information for routing. Principles of geometric routing approaches are very simple. Every node is assumed to be aware of the location of itself, of its neighbors, and of the destination. Based only on these information, every node is able to perform a routing decision. The location can be determined by either geographic coordinates (we thus talk of geographic routing) or logical coordinates extracted from the environment. In the former case, location coordinates may be derived thanks to GPS or estimated thanks to any other positioning mean such as triangulation. In the latter case, a new coordinate system has to be built. This chapter reviews the main routing algorithms in every coordinate-based system, highlighting the strengths and weaknesses of each of them.

15.1 Introduction

Routing in wireless sensor networks is a challenging task. Many different approaches have been proposed in the literature. We can identify three main classes of routing protocols: (i) proactive routing such as OLSR [10], (ii) reactive routing such as AODV [29], and (iii) geometric routing or georouting. This latter approach is receiving more and more attention since it is a memoryless and scalable approach, unlike the two other ones. In addition, it better suits the constraints of wireless sensor networks in which memory and processing capacities are very low and in which the number of entities is expected to be very high. The idea of using position information for routing was first proposed in the 1980s in the area of packet radio networks and interconnection networks.

N. Mitton (✉)
Centre de recherche INRIA Lille - Nord Europe, Lille, France
e-mail: nathalie.mitton@inria.fr

Principles of geometric routing approaches are very simple. Every node is assumed to be aware of the location of itself, of its neighbors, and of the destination. Based only on these information, every node is able to perform a routing decision. The location can be determined by either geographic coordinates (we thus talk of geographic routing) or logical coordinates extracted from the environment. In the former case, location coordinates may be derived thanks to GPS or estimated thanks to any other positioning mean such as triangulation [2, 27]. In the latter case, a new coordinate system has to be built. Coordinates are said as *virtual coordinates*. Yet, geometric routing is

- localized: only local information such as the position of the current node holding a packet, the one of its neighbors, and the one of the destination is required to take a routing decision. Localized algorithms avoid communication overhead, which yields a scalable protocol;
- distributed: every node performs the same algorithm;
- memoryless: no additional information has to be stored neither on the nodes on the path nor in the message;
- scalable.

Indeed, unlike traditional routing schemes (either proactive or reactive), georouting does not need to flood the whole network and does not store any routing tables. These features make them more scalable (in terms of memory and bandwidth overhead) and more energy efficient (since no useless message is sent to discover routes).

Each of both families of georouting protocols (either with exact or with virtual coordinates) can be divided based on its properties with respect to the metric used (hop count or power) and whether or not it guarantees delivery. Therefore, there are four classes of algorithms: (i) simple hop count-based algorithms without guaranteed delivery, (ii) hop count based with guaranteed delivery, (iii) energy efficient without guaranteed delivery, or (iv) energy efficient and guaranteed delivery.

In this chapter, we review the different geometric routing protocols from the literature with respect to the kind of coordinates they use. Section 15.2 reviews the propositions based on geographic coordinates (latitude, longitude, altitude) while Sect. 15.3 focuses on protocols based on virtual coordinates. In each of these sections, we will see different approaches that have been proposed to achieve energy efficiency and packet delivery. In addition, Sect. 15.3 reviews several ways proposed to provide the virtual coordinate systems, highlighting strengths and weaknesses of each one. Finally, Sect. 15.4 summarizes the different solutions.

15.2 Geometric Routing Based on Geographic Coordinates

In this section, we consider that nodes are aware of their geographic coordinates. Like already mentioned, in such protocols, the routing decision is performed by a node holding a packet only based on the information of the position of itself, of its neighbors, and of the destination node. These positions may be exact if retrieved from a positioning device (GPS or Galileo). If only a part of nodes are equipped with a positioning device, geographical coordinates may be estimated through triangulation [27] or any other mean [6], based on the neighborhood tables and the coordinates of GPS nodes.

In order to cope with the lack of positioning information and take benefit from georouting paradigms, some works have proposed to evaluate the distance between each node in order to approximate the relative geographical position of the node in a two-dimensional plane. The approach proposed in [6] uses techniques such as RSSI (receive signal strength information), TOA (time of arrival), AOA (angle of arrival), or TDOA (time difference of arrival) [28] to estimate the distance between two nodes. Based on regular packet sending, every node knows its one-hop and two-hop neighborhood and knows some of the distances between its one-hop and two-hop neighbors. First, each node i chooses two nodes j and k from its one-hop neighbors that do not lie on the same line and with a known distance greater than 0. Node i then defines a local coordinate system based on i, j, and k. It holds coordinates $(0, 0)$. Using triangulation, each neighbor of node i can be positioned based on this local coordinate system. In a second phase, all local coordinate systems are modified by rotation or mirroring to achieve the same direction. In the third phase, an election algorithm is applied to choose the center of the network coordinate system and this network coordinate system is broadcast in the network. As a result, we obtain a relative and approximate (due to distance approximation) positioning system which is used exactly as coordinates provided by a GPS.

The position of the neighbors may be achieved through periodic exchange within a neighborhood where every node broadcasts its position in Hello packets. The position of the destination may be achieved through a localization algorithm like [6, 30, 40], but the review of localization scheme is beyond the scope of this chapter.

15.2.1 Greedy and Directional Approaches

First routing approaches were pretty simple and intuitive. For instance, in [26], the node holding a packet has to choose at random a forwarder among the neighbors in the forwarding direction of the destination. Yet, in Fig. 15.1, s will choose at random between nodes in the gray area, i.e., a, b, c, or e. Choosing the next hop only among the nodes in the forwarding direction guarantees that at each step, a progress is made toward the destination and no loop is created.

Then, in the greedy method [14], a node s holding a packet forwards it to its neighbor a that is the closest to the destination d. Greedy forwarding tries to bring the message closer to the destination at each step using only local information, aiming at reducing the overall number of hops. Thus, each node forwards the message to the neighbor that is most suitable from a local point of view. The most suitable neighbor in the "Greedy case" is the one who minimizes the distance to the destination at each step. In Fig. 15.1, node s sends its message to node a.

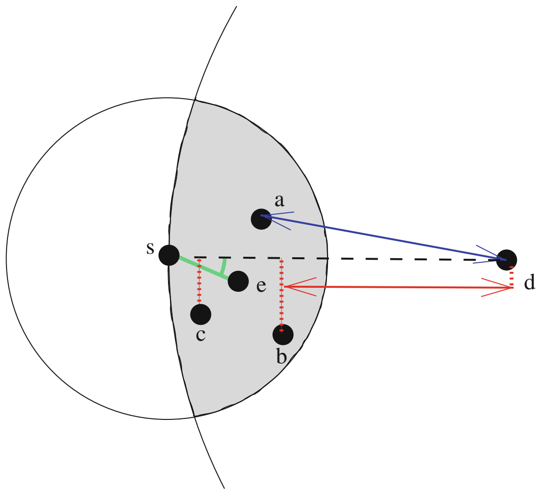


Fig. 15.1 Illustration of geographic greedy routings

Alternatively, one can consider another notion of progress. MFR [35] (most forward routing) and NFP [16] (nearest with forwarding progress) consider the projected distance on the source–destination line. In MFR, the sending node selects as the next forwarder the node with maximum projected distance while NFP selects the one with the smallest one. MFR tries to get closer to the destination at each step by sticking to the initial direction in order to limit the number of hops needed to reach the destination. NFP suggests to adjust the transmission power to the distance between the two nodes in order to save energy during the transmission. In Fig. 15.1, MFR would select node b while NFP would designate node c as the next forwarder.

Another alternative considers the neighbor which provides the minimum angle between source–neighbor and source–destination. This approach is known as DIrectional routing (DIR) or compass routing [18]. With such an algorithm, node a selects node e in Fig. 15.1. The idea here is to stick to the direct direction in order to reduce the stretch factor of the resulting routing path.

Basagni et al. incorporate mobility concerns by introducing DREAM [1]. Indeed, the source node first determines an angular sector for forwarding, based on the mobility information of the destination node d. Then, the message is forwarded to every node lying in that angular sector. To determine this angular sector, node s computes the circle centered on d with radius equal to the maximum possible movement of d since the last update. Then, the angular sector is defined by the tangent to that circle passing by node s. For instance, in Fig. 15.2, a forwards the message to every node in the gray area, i.e., nodes a and b. A very similar approach has been proposed in parallel than DREAM in the same conference. It is called LAR, for location-aided routing [17], and presents only few modifications. Here, if there is no node in the computed angular sector, this sector is enlarged till including one.

For all these methods, if the routing ends up at a node which has no neighbor closer than itself to the destination, the routing fails. In such a case, Finn [14] pro-

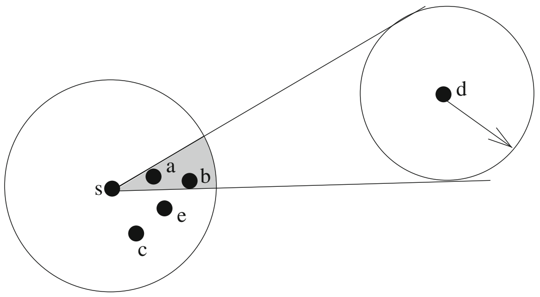


Fig. 15.2 Illustration of DREAM

poses to search all n-hop neighbors (nodes at distance at most n hops away from the current node, where n is a network-dependent parameter) by flooding the nodes until a node closer to destination than the current node is found. The algorithm has non-trivial details and does not guarantee delivery nor optimize flooding rate.

A variant of greedy algorithms, called GEDIR, is proposed in [33]. In this variant, the message is dropped if the best choice for a current node is to return the message to the node the message came from. It increases delivery rate by prolonging failure. The same criterion can be applied to MFR method and directional methods. GEDIR is often used as the basic ingredient in other routines. For instance, it is used in several location update schemes, such as quorum-based and home agent-based schemes.

Globally, basic greedy strategy based on distance is loop free. Indeed, at each step, the message has to move forward to the destination and thus cannot loop by going through a node it has already visited. MFR has also been proved to be loop free [33] while DIR is not loop free as shown in Fig. 15.3. Indeed, in Fig. 15.3, let us suppose that nodes a and b are not neighbors. By applying DIR, node s forwards to node a. c is the best choice for a as the next forwarder since it is the node which minimizes the angle toward d, which makes the message go backward. A loop then appears.

All the greedy approaches described to that point are based either on the distance or on the direction. None of them but NFP has energy consumption concerns. As already mentioned, NFP tries to minimize the energy consumption by sending the

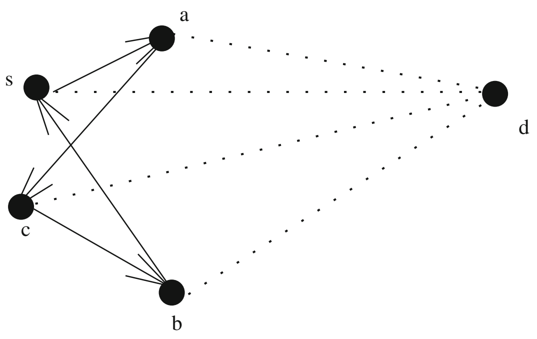


Fig. 15.3 DIR is not loop free

message to the closest node to a in the direction of the destination. This leads to a succession of small hops, less energy consuming than long hops. Nevertheless, generally, the energy consumed depends indeed not only on the transmission range r but also on the overhead c due to signal processing. The most commonly used energy model is

$$\text{cost}(r) = \begin{cases} r^\alpha + c & \text{if } r \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{15.1}$$

where α is a real constant greater than 1 that represents the signal attenuation. Yet, a succession of small hops is not necessarily less energy consuming than a succession of a smaller number of greater hops.

In [34] the optimal transmission radius, r^*, that minimizes the total power consumption for a routing task is computed and it is equal to

$$r^* = \sqrt[\alpha]{\frac{c}{\alpha - 1}} \tag{15.2}$$

Based on these observations, the first power-aware localized routing algorithms were described in [34]. Cost-over-progress (COP) framework with power as the cost has been applied in [19]. Let us take Fig. 15.4 to illustrate it. To forward a packet to destination node d, source node s considers only nodes in the forwarding direction of d (nodes in the gray area). It selects among them its neighbor a such that the ratio of the energy consumed to reach that neighbor (cost($|sa|$)) to the progress made (measured as the reduction in distance to d, i.e., $|sd| - |ad|$) is minimized. The idea is the following. Ideally all hops from nodes s to d provide the same progress as the first one via candidate neighbor a. The number of such hops along the path from s to d is then $|sd|/|sd| - |ad|)|$, and cost of each is cost($|sa|$). $|sd|$ being a constant, at each step, the algorithm tries to optimize the ratio cost($|sa|$)$/|sd| - |ad|$.

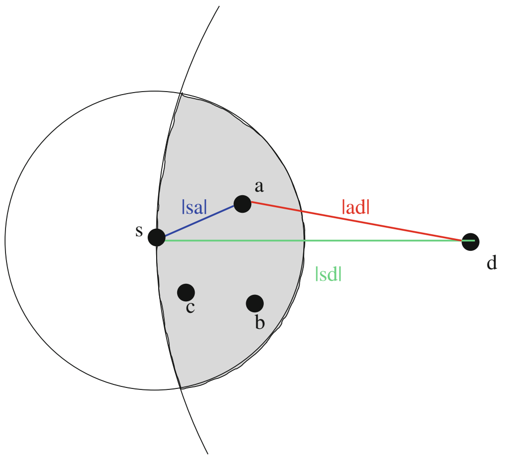


Fig. 15.4 Illustration of COP

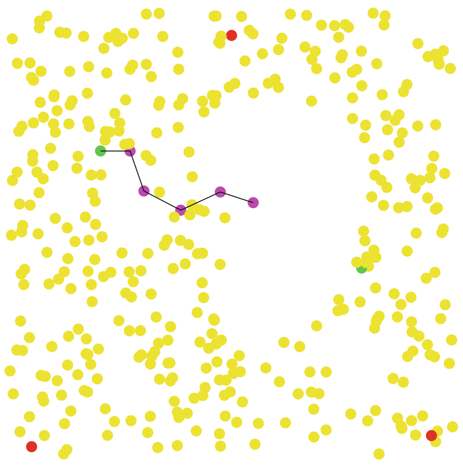


Fig. 15.5 Illustration of a failure routing. Source and destination nodes are the green nodes. Nodes on the routing path appear in *red*. Routing reaches a coverage hole and fails

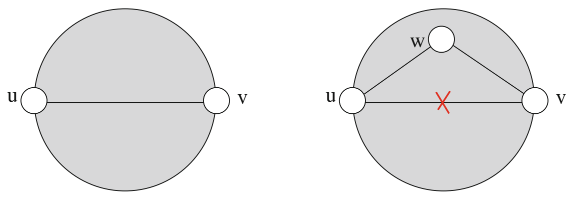
The same paper [19] proposes and analyzes another algorithm. The iterative power progress algorithm is an improvement of the basic COP algorithm. It works as follows. As in COP, a node s, currently holding a message destined to d, first finds a neighbor a that minimizes $\text{cost}(|sa|)/(|ds| - |da|)$. Then, the search continues for an intermediate node b which (i) is closer to d than s, (ii) is neighbor to both s and a, and (iii) satisfies $\text{cost}(|sb|) + \text{cost}(|ba|) < \text{cost}(|sa|)$ and has the minimum $\text{cost}(|sb|) + \text{cost}(|ba|)$ measure. If found, such node b replaces a as selected neighbor, and the search for a better intermediate node repeats. This process is iteratively repeated until no improvement is possible, and node s forwards the message to the selected neighbor, which then applies the same scheme for its own forwarding.

Till that point, we have reviewed most of the greedy routing protocols, hop count based or energy aware. Though these routing algorithms work well in dense networks, they fail if the node holding the message is closer to the destination than any of its neighbors. Indeed, in sparse graphs, these algorithms suffer from coverage hole and may fail, as Fig. 15.5 illustrates. Therefore, some studies introduce solutions that guarantee delivery.

Yet, several propositions have been proposed in the literature for greedy routing in wireless networks, with energy concern when the radius range can be adapted on demand by nodes. Nevertheless, none of these approaches guarantee the packet delivery even if the network is connected. Therefore, more investigations have been performed on this track.

15.2.2 Guaranteed Delivery Approaches

In order to guarantee the message delivery, the authors of [4] have proposed the face routing. Face routing guarantees delivery in two-dimensional UDG. Face routing requires the network topology to be a planar graph (i.e., no edges intersect each other). To planarize a graph, several algorithms can be used such as Gabriel



(a) Case edge *uv* belongs to GG.

(b) Case edge *uv* does not belong to GG.

Fig. 15.6 Illustration of Gabriel graph building. In (**a**), there is no node lying in the disc with diameter $[u, v]$, edge uv belongs to the GG. But, in (**b**), the presence of node w in the disc removes edge uv from GG

graph(GG) [4, 20], the relative neighborhood graph (RNG) [36], or the Morelia graph [3]. Gabriel graph, for instance, contains edges between nodes u and v if and only if no other nodes are located inside the circle of diameter $|uv|$, as illustrated in Fig. 15.6. GG has some desirable properties when used for routing in wireless networks such as localized message, free computation, planarity, and preserving connectivity [4]. If the initial underlying graph is a graph $G = (V, E)$, the outcome planarized graph is a subgraph $G' = (V, E') \subset G$ where every intersecting edge has been removed.

Planarization divides the network into faces. Face routing then performs as follows on the planarized graph G'. The face that contains the line (sd), where s is the source node and d is the destination node, is traversed by right-hand or left-hand rule (placing a virtual hand on the wall of the face). When edge which has to be followed intersects with the imaginary dash line (sd), the message changes face, and so on, till reaching the destination node. Figure 15.7 illustrates the face algorithm. To send a message to node d, s follows the first face (composed by nodes $sabgh$) and forwards it to node a. a applies the same algorithm and forwards to b. If b continued to follow the same face, it would send the message to node g and by doing so would cross red line (sd). Therefore, b changes face and sends the message to node c. The same algorithm is applied by every node on the path. Finally, the message follows blue edges in figure, going through nodes $sabciefd$.

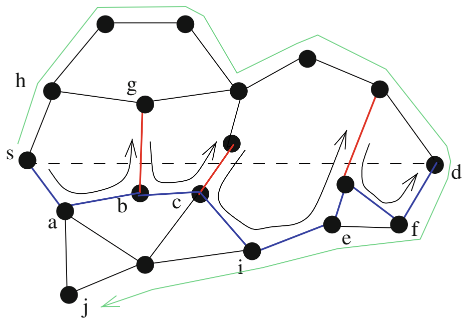


Fig. 15.7 Illustration of face routing. Path from s to d appears in *blue*, path from s to j in *green*

Theorem 1 *FACE guarantees the packet delivery.*

Proof We give a sketch of proof of a variation of the FACE routing algorithm. This version of FACE is the following. We consider the current node s with the destination node d. At each step, the current node s knows the position of the destination d and has in input a reference point r which is used to decide the face traversal and the change of face. At the starting point, we take $r = s$. The current node s selects the face which contains the segment $[rd]$, and the message starts the face traversal according to the right-hand or left-hand principle. The change of face is decided when the message crosses the segment $[rd]$ by using an edge (b, g) (by using Fig. 15.7 notations). In this case, we set $r =]rd] \cap [bg]$ and $s = b$. We can note that it has been shown that we can also apply $s = g$ without real impact on the correctness of the algorithm or on the performances of the routing protocol. After changing face, we apply the same algorithm in order to select the next face.

If at each face traversal we find an edge (b, g) which crosses $]rd]$, we can observe easily that the distance $|rd|$ is strictly decreasing and then the delivery is guaranteed. Then, without loss of generality, it is enough to show that the edge $]rd]$ is always crossed in the face traversal. This is equivalent to show that there always exists an intersection between the segment $]rd]$ and the selected face.

We distinguish two cases: the selected face is an internal face or the selected face is an external face of the graph of the whole network. In the case of internal face, the crossing of the segment is guaranteed if the destination node is not isolated in the middle of the selected face since the selected face contains the segment. This case cannot happen if the network is connected. In the case of external face, if the destination node is not part of the external face, the existence of an intersection between the segment and the face is sure since a part of the external face is situated under the $]rd]$ segment and another part is situated above the segment. Indeed, the segment $]rd]$ "enters" in the graph at a point A which is at the intersection of the face and the segment. If the destination node D is part of the external face, the existence of the intersection is trivial. ■

Yet, face routing guarantees message delivery. But using only face routing may generate very long paths in cases where the message has to follow the external face of the network. This is, for instance, the case in Fig. 15.7 when node s needs to reach node j. The message follows the green line. Therefore, to overcome this drawback and to take advantages of both greedy and face solutions, the authors of [4] propose the GFG (greedy–face–greedy)-based approach. It applies greedy routing until either the message is delivered or the routing fails. In the latter case, face routing is applied to recover from failure. It has been shown in [15] that face routing guarantees recovery traversing the first face. Greedy routing continues from a until delivery or another failure node is encountered. Through guaranteed delivery and providing path with a fairer stretch factor than face, GFG remains not energy efficient.

The first tentative to address guaranteed delivery in power-aware localized routing is [32]. It is a greedy–face–greedy (GFG) approach where greedy routing is the COP as in [34] while face routing is similar to the one in [4]. One of the drawbacks of face routing is that it is likely to follow a long sequence of short edges of GG. Although short edges are less energy consuming than long edges, a succession of short edges will be more energy consuming than a short sequence of medium edges. Yet, for sparse networks where the face step is often triggered, this approach is not energy efficient.

In LEARN [39], a localized energy-aware routing is proposed. LEARN assumes that every node is aware to adapt its transmission range. A node s aiming at destination d selects neighbor b inside a restricted neighborhood ($\widehat{bsd} \leq \alpha$ for a parameter $\alpha < \pi/3$) that has the largest energy mileage, determined as the ratio $|sb|/\text{power}(|sb|)$ where $\text{power}(|sb|)$ represents the cost to send a message from node s to node b. If no such neighbor exists inside the restricted neighborhood, LEARN fails. In the variant LEARN-G, a node switches to greedy routing [14] in case of failure and selects the neighbor closest to the destination. Finally, in the variant LEARN-GFG, a node invokes face routing when a failure occurs. Thus, as previously, LEARN can be energy efficient only when the network is dense enough and that every node on the path may find a neighbor of it in the α angular sector toward the destination (no invocation to greedy or face procedures).

The authors of [39] show that when LEARN indeed finds a path without needing to invoke neither greedy nor face routing, the total Euclidean length of the final path from s to d is within a constant factor of the optimum direct path (see Theorem 2) and that this path is energy efficient. These proofs have then been generalized and extended in [12] to any protocol, providing paths meeting angular constraints as follows:

Definition 1 A path meets angular constraints if every hop is within an angle $\theta \leq \alpha < \frac{\pi}{3}$, $\theta \to 0$ toward the destination.

Theorem 2 (Wang et al. [39]) *Any path from s to d meeting angular constraint has length that is a constant of the optimum $|sd|$.*

Proof The proof is made by induction on the number of hops. The theorem is clearly true when the path has only one hop (the path has optimal length). Assume that it is true for the path with $(k-1)$ hops. Then consider any k-hop path $v_0v_1v_2 \ldots v_{k-1}v_k$. By induction, the length of path $v_0v_1 \ldots v_{k-1}$ is at most a constant δ from the optimum: $\sum_{i=1}^{k-1} |v_{i-1}v_i| = \delta \times |v_0v_{k-1}|$. Then it is sufficient to show that $|v_{k-1}v_k| + \delta \times |v_0v_{k-1}| \leq \delta \times |v_0v_k|$.

Let us consider triangle $v_0v_1v_k$, in which, by the routing protocol, v_0 is the longest link and $\angle v_1v_0v_k = \theta < \alpha$ (angular constraints). Let us note $\angle v_0v_kv_1 = 2x$. Then a simple geometry computation shows that $\frac{|v_0v_1|}{|v_kv_0|-|v_kv_1|} = \frac{\sin(x+\frac{\pi}{2})}{\frac{\pi}{2}-x-\theta} = \frac{\cos x}{\cos(x+\theta)}$. This means that we need $x < \frac{\pi}{2} - \theta$ and $x < \frac{\pi-\theta}{4}$. Simple computation shows that $\frac{\cos x}{\cos(x+\theta)} < \frac{1}{1-2\sin\frac{\theta}{2}} < \frac{1}{1-2\sin\frac{\alpha}{2}}$ for $x < \min\left(\frac{\pi}{2} - \theta, \frac{\pi-\theta}{4}\right)$. And thus $|v_{k-1}v_k| + \delta \times |v_0v_{k-1}| \leq \delta \times |v_0v_k|$ for every routing protocol respecting angular constraints. ■

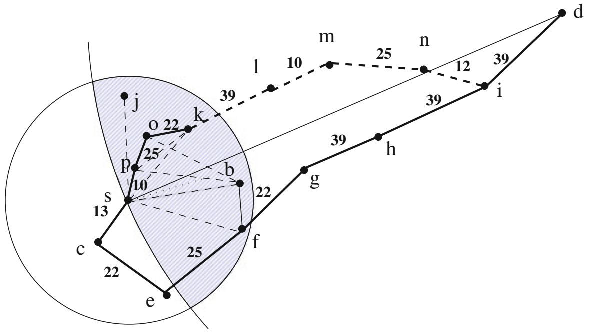


Fig. 15.8 Illustration of the greedy step of the SPFSP and EtE. *Dash links* are the links with high weights

In [31], SPFSP (shortest path face shortest path), a GFG-based energy-aware routing with guaranteed delivery, has been proposed. The energy awareness is introduced via the computation of an energy-weighted shortest path (with power consumption as weights) at both greedy and face steps. Let us take the example plotted in Fig. 15.8 to illustrate this greedy step. Node s currently holding a packet first selects a target node by using the plain greedy algorithm [14], i.e., its closest neighbor to destination d: node b in figure. Instead of transmitting directly to b, s computes the energy-weighted shortest path to b over its whole neighborhood. In the figure, this path is $scefb$. Indeed, node s also considers its neighbors that are not in the forwarding direction of d like nodes c and e. This path is then followed by the packet until reaching node f, which is the first node on the path closer to d than s. Node f then recursively applies the same protocol till either reaching the destination node d or to fail. Note that, to ensure no loop, the shortest path has to be embedded in the message, creating an overhead. To recover from failure points, a face routing [4] is used in the following way. If we suppose that node s in Fig. 15.7 is the failure node, it applies face routing only to determine the target node: node a in Fig. 15.7. But, once again, instead of reaching that target node directly, node s computes an energy-weighted shortest path over its neighborhood and reaches node a via this shortest path. Nevertheless, simulations have shown that most of the time, this enhancement added to the recovery step is of no use since the shortest path is most of the time the edge of the face itself. This is due to the fact that GG keeps only small edges in the underlying graph and, thus, face edges are among the smallest ones and thus among the less consuming edges within a node neighborhood.

Figure 15.9a illustrates a sample execution of the SPFSP algorithm. Greedy routing proceeds from node 1 which first chooses its next forwarder among nodes 2, 19, and 21 (node 23 is not included in the selection since it is further from the destination than node 1). Node 1 selects node 19 as temporary destination since it provides the best progress toward destination and sends the packet to node 23, the first node on the shortest path toward node 19. Note that even if node 23 was not among the potential targets, it may be included in the path to it. Node 23 then forwards to node 2, second node on the path embedded in the packet by the source node. Node 2 being closer to node 8 than the source, it performs the selection algorithm and finds node 19 as its best forwarder and in this case, the shortest path in that link. Node

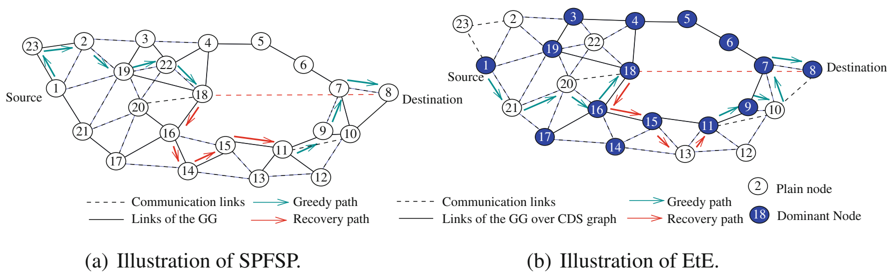


Fig. 15.9 Comparison of SPFSP and EtE. Continuous links are the links to be followed in the recovery step (GG). GG is built over every node in plain face routing or in SPFSP (**a**), while it is computed only over dominant nodes (*blue nodes*) in EtE (**b**)

19 selects node 18 by following a shortest path through node 22 to which it sends the message. The latter then forwards to node 18 where greedy routing fails. Face routing is then invoked to follow edges 18-16 (directly), 16-14, 14-15, and 15-11. Greedy routing then continues till final delivery.

Based on these observations, the authors of [12] have proposed EtE (end-to-end) protocol which guarantees the packet delivery with energy concerns at both greedy and face steps. EtE draws its inspiration from SPFSP. The greedy step is modified in two ways: (i) in the way the selected target is chosen and (ii) on the set of nodes over which the shortest path is computed. Indeed, in order to avoid to embed the path in the packet, the shortest path is computed only on nodes in the forwarding direction of the destination d. In Fig. 15.8, s computes the shortest path only over nodes in the blue dash area, i.e., b, f, j, k, o, and p. The selection of the target node is modified as follows. Instead of selecting the closest node to the destination, node s selects its target node in a *cost-over-progress* fashion [19] where the cost is the cost of the energy-weighted path from node s to the considered node u. Let $v_0 v_1 \ldots v_i v_{i+1} .. v_n$ be the nodes on the shortest path from s to k with $v_0 = s$ and $v_n = u$. The cost of the shortest path $\text{cost}_{\text{SP}}(s, u)$ from s to u is defined as

$$\text{cost}_{\text{SP}}(s, u) = \sum_{i=0}^{n-1} \text{cost}(|v_i v_{i+1}|)$$

Then, node s selects node b which minimizes the cost of the shortest path from s to u divided by the progress it makes toward destination node d.

The target node k is then the one which is such that

$$\frac{\text{cost}_{\text{SP}}(s, k)}{||sd| - |kd||} = \min_{u \in N_d(s)} \frac{\text{cost}_{\text{SP}}(s, u)}{||sd| - |ud||}$$

Once that node is determined, node s forwards the message to the first node on the shortest path from s to k, i.e., node p in figure. Then, node p reiterates the same process. No loop is possible since the next hop is computed only over nodes in the forwarding direction of the destination. This algorithm goes on till the destination is reached or there is no node closer to the destination than the one currently holding the message. In this latter case, an energy-efficient face routing is applied for recovering from failure.

As already mentioned, regular face routing guarantees delivery but is not energy efficient since it may use very short edges compared to the energy-optimal transmission length r^* since GG removes longer edges. To overcome this drawback, the authors of EtE have introduced an energy-efficient variant to face routing. For it, they add a step in the planarization of the graph. From the original graph $G = (V, E)$ (Fig. 15.10a) they compute a connected dominating set (CDS), V' of V. Since source s and destination d may not be in the CDS, the set V' is expanded with s and d. Let $G' = (V', E') \subset G$ where $V' \subset V$ is the expanded set of dominant nodes and $E' \subset E$ is the set of edges between nodes in V'. Then, a CDS election protocol is applied on V'. The authors use the CDS election protocol introduced in [5], but any other election protocol may be applied. Since face routing must be applied on a planarized graph, the Gabriel graph $G'' = (V', E'')$ is extracted from G', where $E'' \subset E'$ is the set of edges remaining in the planarized graph (Fig. 15.10c). Face routing is lastly run over G''. This face routing guarantees delivery in the constructed subset since it contains source and destination nodes and preserves connectivity. Moreover, by considering only edges connecting two dominating nodes, the routing process avoids to choose too short edges. Each node needs to know its neighbors that are in the CDS.

Based on G'', the same principles of the recovery step of [31] are applied. Current node s that is in recovery mode applies face routing on G'' only to decide on which edge (s, b) to follow to reach the destination node from a given node s. However, that edge does not need to be selected since it may be too long ($|sb| > r^*$). Node b is thus reached through an energy-weighted shortest path. If b is closer to the destination node than the node which has initiated the face routing step, node b selects the next hop in the routing path following the greedy routing described above. Otherwise, it determines the node following face routing over CDS nodes and computes the energy-based shortest path to reach it.

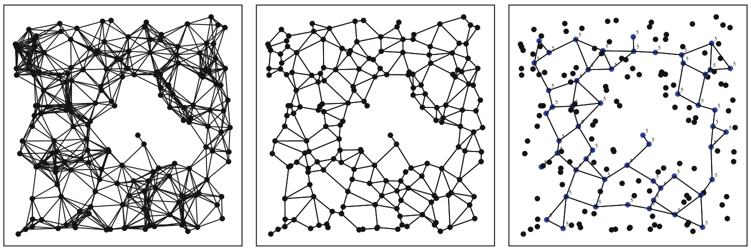

(a) Initial graph G (b) GG on G (c) GG from G'

Fig. 15.10 When using regular face routing over graph G (**a**) (like SPFSP), messages follow GG edges of G (**b**). Edges of G'' (**c**) are used instead when applying the energy-efficient face routing of EtE

Figure 15.9b illustrates a sample execution of EtE algorithm. Greedy routing proceeds from node 1 which first computes the cost of the shortest path toward nodes 2, 19, and 21. Node 1 selects node 19 as temporary destination since it provides the lowest cost over progress and sends the packet to node 21, the first node on the shortest path toward 19. (Node 23 is not included in the computation since it does not provide any progress to destination.) Node 21 finds node 20 as its best forwarder and in this case, the shortest path is that link. Node 20 selects node 18 by following a shortest path through node 16 to which it sends the message. The latter then forwards to node 18 where greedy routing fails. Face routing is then invoked to follow edges 18-16 (directly), 16-15 (directly), and 15-11 (which is replaced by path 15-13-11 for energy efficiency). Greedy routing then continues till delivery to 11 selecting 10 via 9, 9 selecting 10, and 10 selecting destination 8 and delivering via node 7.

Yet, we have reviewed most of the geographic routing for wireless sensor and ad hoc networks in the literature. Even if along the years the solutions proposed are better and better, the ones of them that guarantees delivery all assume a unit disc graph. Indeed, if this assumption does not hold, there is no way to planarize the graph, and thus face routing and variants cannot be applied anymore. Unfortunately, last experiments have shown that the UDG does not hold in a realistic network. Yet, the new challenge appearing here is to investigate geographic routing solutions that both are energy efficient and guarantee delivery but in any arbitrary graph (not only unit disk graph).

15.2.3 Anycasting

In the anycasting problem, a sensor wants to report event information to one of sinks or actors. The authors of [25] describe the first localized anycasting algorithms that guarantee delivery for connected multi-sink sensor and sensor–actor networks.

Three geographic anycast algorithms are proposed: GFGA, COPA, and EEGDA which are inspired, respectively, by GFG [4], COP [34], and EtE [12]. Each of them consists of greedy and recovery phases, and they all guarantee delivery for a report from a sensor if it is connected to at least one sink or actor. GFGA uses hop count as the metric, while others apply power consumption, where both greedy and recovery steps are energy efficient. The two energy-efficient algorithms have different computing complexities.

All algorithms construct a path from the source sensor node to one of sinks/actors. During the path construction, there exists a single destination to reach. The main feature of these algorithms is that this destination may change along the path of the message according to the network topology. Anycasting may start from current node s toward sink/actor $S(s)$ that is the closest to it. However, s could in fact be even disconnected from $S(s)$ or closer in number of hops to another sink.

In greedy phase, and with hop count as the metric, i.e., GFGA, node s currently holding the packet forwards it to its neighbor v that minimizes $|vS(v)|$ (is closest to its nearest actor/sink). When an arbitrary cost metric is used (COPA), the selected neighbor v minimizes the ratio of cost cost($|sv|$) of sending packet to v over the reduction in distance ($|sS(s)| - |vS(v)|$) to the closest actor/sink. An improved variant, EEGDA, is to forward to the first neighbor on the shortest weighted path toward v instead of sending directly to v, like in EtE. If none of neighbors reduces that distance then recovery mode is invoked. It is done by face traversal toward the nearest connected actor/sink, where edges are replaced by paths optimizing given cost. A hop count-based (FACE-like) and two variants of localized power-aware (Ete-recovery-like) anycasting algorithms are described.

15.3 Virtual Coordinate Systems

As noticed in previous sections, georouting solutions are very promising solutions for wireless sensor networks since coordinates simplify the routing decision at each node by limiting bandwidth and memory overheads. Nevertheless, geographical position information provided by devices such as GPS or Galileo is not always a feasible solution. Indeed GPS-like positioning systems are bulky, energy costly, and expensive and are not adapted for every environment [28]. Therefore, literature has witnessed the birth of protocols that assign "virtual" coordinates to each node to take benefit from georouting techniques.

Indeed, it is worth noting that virtual coordinates do not necessarily need to embed global positioning information, and they just have to be consistent enough to allow georouting. Two well-spread methods have been proposed. The most used positioning technique is based on hop count distance from given landmarks and is described in Sect. 15.3.1. Many routing protocols are based on this virtual coordinate system. They mainly differ in the distance functions and routing progress they use, as we will see. Though simple, such a landmark-based system exhibits some drawbacks due to coordinate constructions that we will detail. Therefore, a new tree-based coordinate system has been proposed (Sect. 15.3.2).

15.3.1 Landmark-Based Coordinate System

Landmark-based coordinate system is based on hop distance between the sensors and some specific nodes and do not try to approximate physical coordinates. Therefore, the virtual topology can be unrelated to the physical topology of the network.

15.3.1.1 Landmark-Based Coordinate System Construction

The landmark-based coordinate system is built into two steps: (i) flooding by landmarks and (ii) computation of coordinates. The first step is common to every landmark-based georouting protocol while the second one differs from one to another.

The first step can be split into two phases. In the first phase, a global and distributed election mechanism elects a set of nodes as landmarks or anchors. Nodes acting as landmarks can be explicitly designated by an external process at the bootstrap of the network. During the second phase, every landmark floods a message containing a counter which is incremented at each hop. In the sequel, the term "broadcast" stands for message propagation in a node's neighborhood and the term "flooding" refers to network-wide message propagation. At the end of this second phase, every arbitrary node i can thus determine a vector $V(i) = \left(h_1^i, ..., h_n^i\right)$ where n is the number of landmarks and h_n is the hop distance between node i and each anchor node (node i is h_1^i hops away from landmark 1).

The second step allows every node i to compute its virtual coordinates based on vector $V(i) = \left(h_1^i, ..., h_n^i\right)$. To do so, different functions can be used depending on the protocol. The virtual coordinates of node i are $X(i) = \Gamma(V(i))$ where $X(i) = \left(x_1^i, ..., x_m^i\right)$ and $m \leq n$ where different Γ functions are used in the literature. The most common Γ functions are the following ones:

- The "identity" function denoted by Γ_{id} where $X(i) = V(i)$ with $m = n$. This is the simplest function used, for instance, in VCap [2, 7] or VCost [11].
- The "centered virtual coordinates" function [13] denoted by Γ_{cvc} and $x_j^i = \left(h_j^i\right)^2 - \mu$ for $m = n$, $j = 1, \ldots, n$ where $\mu = \frac{1}{n}\sum_{j=1}^{n}\left(h_j^i\right)^2$.
- The "averaging" function [23] denoted by Γ_{av} gives the following relationship between $X(i)$ and $V(i)$:

$$x_k^i = \frac{\frac{\sum_{t=1}^{|N(i)|} h_k^t}{|N(i)|} + h_k^i}{2}$$

where $|N(i)|$ is the number of neighbors of node i.

All these Γ functions are used to compute the virtual coordinates of nodes based on their hop distance to every landmark. As next section will show, they all exhibit different features and impact the routing process. Figure 15.11 illustrates the results of coordinate assignment with identity Γ function (Γ_{id}) such as in VCap [7] and/or JUMPS [2]. In this figure, landmarks (anchors) are arbitrarily chosen as nodes L_1, L_2, and L_3. Therefore, every node has a three-dimensional vector as coordinates constituted by the number of hops between itself and every landmark. In this figure, landmark L_3 has coordinates (4, 3, 0) since it is four hops away from landmark L_1 (through nodes 2, 3, and 11), three hops away from landmark L_2 (through nodes 7 and 11), and zero hop away from itself. Since flooding message from each landmark may be received more than once, a node chooses the minimum hop distance to each landmark.

Georouting is then performed on top of $X(i)$ coordinates. In greedy methods, as in geographic routing based on real coordinates, the next hop is the node u that minimizes the distance $\delta(u, d)$ to the destination d (maximal progress). Distance $\delta(u, d)$ is computed over virtual coordinates and once again, several distance functions can be used according to the routing protocol. The most common distances found in the literature are the following:

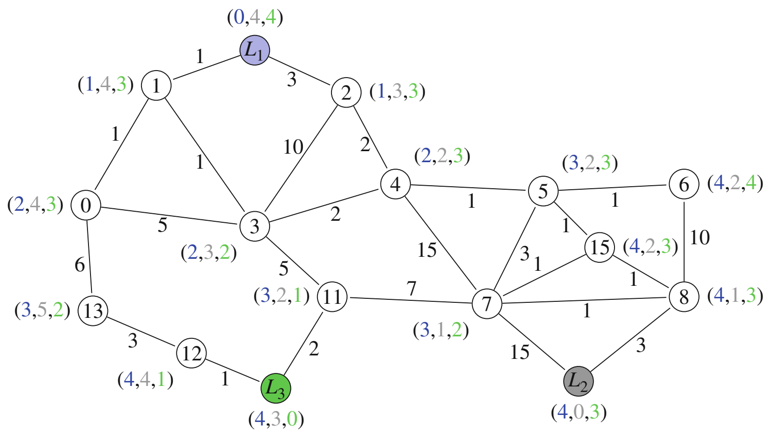


Fig. 15.11 Illustration of hop distance coordinate assignment. In this figure, L_1, L_2, and L_3 are the landmarks. Each node, including landmarks, has a three-dimensional vector as coordinates

- Euclidean distance $\delta_e(u, d) = \sqrt{\sum_{j=1}^{n}\left(x_j^u - x_j^d\right)^2}$,
- Hamming distance $\delta_h(u, d) = \sum_{j=1}^{n}\left|x_j^u - x_j^d\right|$,
- Square Euclidean distance $\delta_s(u, d) = \delta_e(u, d)^2$.

It is worth noting that combining the different Γ and δ functions provides a great amount of different techniques. In the next section, we review the protocols from the literature, explaining what combination they use.

15.3.1.2 Routing on Top of Landmark-Based Coordinates

Even if routing based on landmarks coordinate system may not always be successful, there exists many georouting protocols based on hop distance but their performances differ due to the distance (δ function) and coordinate (Γ function) computation used in each protocol. This section reviews and classifies georouting based on landmark-based system.

The simplest algorithm is VCap. In *Vcap* [7], greedy georouting is performed on top of landmark-based coordinates by combining the Γ_{id} function to compute node coordinates and the δ_e distance function. The system is composed of three landmarks. The next hop is the node u that minimizes the Euclidean distance $\delta_e(u, d)$ to the destination d. $\delta_e(u, d)$ is computed over virtual coordinates as follows: $\delta_e(u, d) = \sqrt{\sum_{j=0}^{n}\left(h_j^u - h_j^d\right)^2}$. To illustrate a routing process, let us assume that node L_2 needs to send a packet to node L_3. To select the next hop, node L_2 computes the distance from all its neighbors to the destination. If the Euclidean distance is used, $\delta_e(7, L_3) = \sqrt{(9)}$, $\delta_e(8, L_3) = \sqrt{13}$, and $\delta_e(L_2, L_3) = \sqrt{18}$. L_2 chooses among its neighbors u with positive progress ($\delta(u, L_3) < \delta(L_1, L_3)$), the closest one to the destination. Since $\delta_e(7, L_3) < \delta_e(8, L_3)$, L_2 chooses node 7 as its next hop. Then, node 7 elects node 11 since $\delta_e(11, L_3) < \delta_e(15, L_3) < \delta_e(5, L_3) < \delta_e(4, L_3) < \delta_e(L_1, L_3)$. As node 11 is a neighbor of node L_3, the greedy routing technique is successful.

JUMPS [2] provides a coordinate system similar to VCap. JUMPS only differs from VCap in the fact that it may use more than three anchor nodes. Similar to VCap, Γ_{id} is chosen as the basis of the coordinate system. Obviously, using these kinds of coordinates may end up to several nodes holding the same virtual coordinates leading to routing ties, which reduces the delivery ratio. Therefore, in addition, JUMPS provides a study on the impact of the landmark placement on routing delivery rate. It happens that the better landmark placement is when landmarks are spread at equal distance one from each other all around the network. This is indeed the placement which reduces the number of nodes holding the same coordinate.

To palliate or at least reduce this drawback and thus increase the delivery rate, the authors of *AVCS* [23] do not use necessarily integer coordinates. Floating coordinates depending on the neighborhood of each node are used instead of hop count. AVCS uses the Γ_{av} which performs a centroid transformation as an averaging function to compute the floating coordinates of the node. The authors of AVCS use the Euclidean distance but with different virtual coordinates. Indeed, the distance between the current node i and the destination node v is $\delta_e(V(v), X(i))$. Results show that a greedy routing on top of AVCS coordinate system outperforms greedy routing protocol on top of geographical coordinates since using virtual coordinates avoids the routing holes. It is worth noting that in AVCS, each node keeps the $V(i)$ and $X(i)$ coordinates which increases memory consumption. The authors also suggest the possibility of applying the Γ_{av} function more than once to reduce the probability of having the same coordinates for two nodes. That is, the Γ_{av} function is applied on $X(i)$ coordinates. Moreover, and in order to even more reduce redundant coordinates, the authors suggest to apply the Γ_{av} function by using the two-hop neighborhood of a node.

In *Gliders* [13] another way of assigning virtual coordinates is described to avoid bad placement of landmarks by taking into account holes in the network. Nodes are partitioned into tiles and landmarks are selected using Voronoi cells [38]; combinatorial Delaunay triangulation is used to estimate the global topology. Virtual coordinates for each node are derived from the node's distance (hop distance) to nearby landmarks by using Γ_{CVC} function. The authors of [13] also describe a routing protocol associated with this coordinate system. In their routing protocol, nodes have to compute a sequence of tiles for inter-tile routing paths, then a gradient descent procedure based on a proper distance function close to the Euclidean distance is used to route packets in a greedy way for intra-tile routing. However, if the coordinate system proposed in [13] can avoid routing ties depending on the density, the routing protocol does not guarantee packet delivery since packet may still reach dead ends. In addition, such a coordinate system associated with such a routing algorithm are very complex and induce a huge memory, bandwidth, and computational overhead, which makes it not scalable and difficultly implemented.

15.3.1.3 Energy Efficiency

In order to increase the delivery rate of georouting protocols using landmark-based coordinates, the authors of *VCost* [11] explore the use of several kinds of Γ and δ functions for coordinates construction and compare their performance. Moreover, they suggest the use of the Hamming distance δ_h instead of using Euclidean distance for routing decision. Results show that the use of Hamming distance increases the delivery rate and can reduce the path length compared to Euclidean distance since Euclidean distance first tries to minimize the maximum difference between coordinates (see Sect. 15.3.1.4). But the main goal of the work presented in [11] is first to provide an energy-efficient georouting on top of virtual coordinates. Therefore, the authors of [11] evaluate and reduce the energy consumption by using a cost-over-progress fashion (see Sect. 15.2.1) to reach the destination on top of virtual coordinates achieved through the Γ_{id} function. Authors assume that nodes are able to tune their transmission range and to estimate the cost of a transmission to each of their neighbors. Nodes select their following next forwarder as the node which minimizes the ratio between the cost of the transmission to the progress provided by this neighbor. The progress is thus computed as the Hamming distance between considered neighbor and destination node. VCost is the first power-aware georouting on top of virtual coordinates.

To illustrate the routing decision of VCost, let us consider the network depicted in Fig. 15.11 and assume a routing from node 4 to node L_2. The cost of each link is given on each edge, and we can see that the cost of the link between node 4 and node 5 is 1. Therefore, we have $\delta_h(L_2, 4) = 4$, $\delta_h(L_2, 5) = 3$, and $\delta_h(L_2, 7) = 3$. The progresses are equal for node 5 and node 7. However, the cost of each link is different, and node 5 is chosen as the next hop of the routing process. The path from node 4 to node L_2 uses nodes 5, 15, and 8.

15.3.1.4 Landmark-Based Positioning System Issues

Yet, routing performed over landmark-based positioning system exhibits some interesting properties. Nevertheless, landmark-based coordinate systems show some issues due to its construction. This section illustrates such particular issues which may reduce the performance of the routing protocol.

Dead End

Let us first consider a packet from node 6 to node 15 in Fig. 15.11. In this case, the greedy technique does not succeed since $\delta_e(6, 15) < \delta_e(15, 5) = \delta_e(15, 8)$ and thus, there is no neighbor of node 6 closest than itself to node 15. This leads to a dead end at node 6. This can arise when using the Γ_{id} function to compute coordinates, but such a situation may also be reached when using other Γ functions. The choice of this function will only impact the number of such situations.

But dead ends can also occur because of another configuration illustrated by Fig. 15.12a. In this figure, node 0 wants to send a packet to node 4. Since, $\delta_e(5, 4) <$

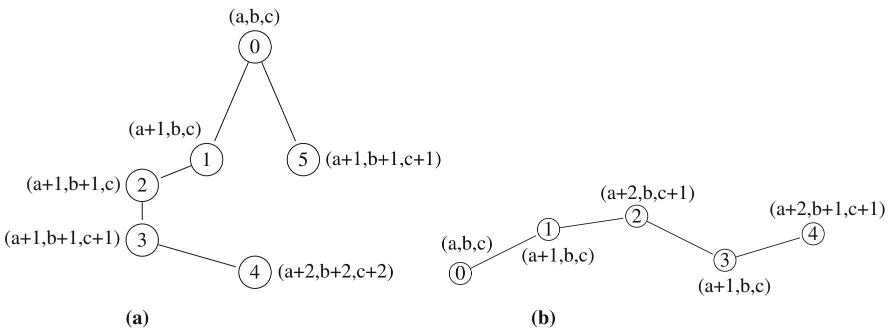


Fig. 15.12 Dead end: (**a**) due to holes and (**b**) due to duplication

$\delta_e(1,4) < \delta_e(0,4)$, node 5 is chosen as the next hop, which leads to a dead end. This latter case is actually the same as the one encountered with greedy algorithms based on geographical coordinates.

Duplicated Addresses

Landmark-based coordinate system may lead to multiple nodes holding the same coordinates. This duplication may have no effect since nodes with the same coordinates are geographically close to each other. However, this is not always the case. Duplication may also lead to dead end since coordinate uniqueness is mandatory to ensure the packet delivery. In Fig. 15.12b, node 3 and node 1 have the same coordinates, therefore, a packet transiting from node 0 aiming to node 4 will go through node 1 and then will stop at node 2 since $\delta_e(2,4) < \delta_e(3,4)$.

Duplicated addresses may also lead to routing ties. Figure 15.13a shows how it can happen. In this figure, node 0 and node 1 hold the same coordinates while they can be geographically far from each other. This duplication is due to the alignment

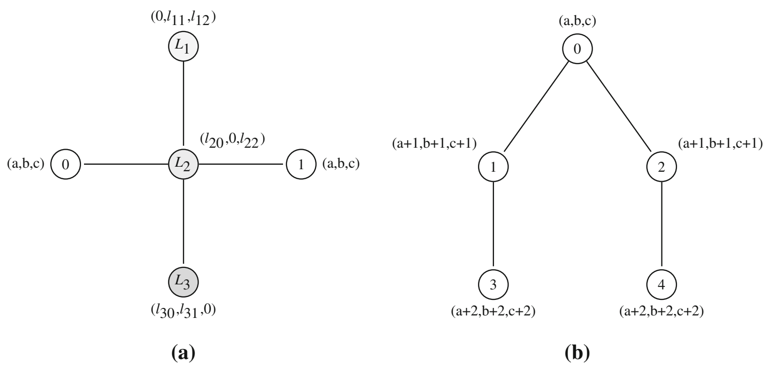


Fig. 15.13 Routing ties: (**a**) bad placement of landmarks and (**b**) one-connected networks

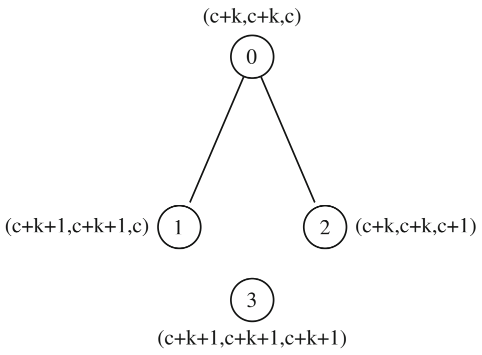


Fig. 15.14 Euclidean versus Hamming distance

of landmarks, which may lead to routing ties. In Fig. 15.13b a part of the network is one-connected, which may lead to the same effect.

Euclidean Versus Hamming Distance

Figure 15.14 shows an example where Hamming distance can reduce the number of hops. In this figure, node 0 wants to send a packet to node 3 with virtual coordinates $(c+k+1, c+k+1, c+k+1)$. When using the Euclidean distance we have $\delta_e(3, 2) < \delta_e(3, 1) < \delta_e(3, 0)$ for $k > 1$. Therefore, node 0 would choose node 2 as its next hop. While considering the Hamming distance, we have $\delta_h(3, 1) < \delta_h(3, 2) < \delta_h(3, 0)$ for $k > 1$; thus, node 0 selects node 1 as its next forwarder. Based on the virtual coordinates, node 1 is the best choice for forwarding since it minimizes the number of hops to reach the destination. For numerical example we can consider $c = 0$ and $k = 2$; thus, we have $\delta_e(3, 2) = 1.73 < \delta_e(3, 1) = 2 < \delta_e(3, 0) = 2.44$ and $\delta_h(3, 1) = 2 < \delta_h(3, 2) = 3 < \delta_h(3, 0) = 4$.

15.3.2 *Tree-Based Coordinate System*

As illustrated so far in this section, although simple and easy to implement, a plain hop-based coordinate system presents some drawbacks, especially regarding delivery ratio. In the following tree-based coordinate systems are developed with associated routing protocols to overcome the issues of classical landmark-based coordinate systems.

15.3.2.1 Tree-Based Coordinate System Construction

A tree-based coordinate system is built into two phases. In the first phase, a global election mechanism chooses a node that acts as the tree root. This node, also called root, initiates the tree construction. Nodes are assigned a unique ID by being labeled either through a depth-first search method or through a breadth-first search method. Routing is performed following the tree. Therefore, routing on top of trees guarantees message delivery since by definition, there exists exactly one path between any pair of nodes. However, the bootstrap phase may be much more complex than in a landmark-based coordinate system according to the protocols. In the following section, the different ways of assigning labels to nodes are explained.

15.3.2.2 Routing on Top of Trees

Liu and Abu-Ghazaleh [22] propose a stateless and guaranteed delivery georouting on tree-based virtual coordinates. They propose a one-dimensional virtual coordinate system based on a depth-first search pre-order traversal of the graph. A tree-style topology is constructed with only connectivity information. Starting from a root node with value 0, which may be randomly chosen, nodes are labeled by sending depth-first search packet to one of their neighbors. The node that receives the packet is assigned a unique identity which is the identity of the packet's sender incremented by 1. If a node does not have any unlabeled neighbor, it sends an end of search packet to its parent. As long as the network is connected, all nodes will eventually receive a unique identifier. Each node m is also given an interval $I(m) = [m, p]$ starting from its label to the greater identity of its children before traversal returns back to its parent. Routing is based on these labels, and current node forwards a packet to the node holding the interval containing the destination. Figure 15.15[1] shows the resulting tree building used for routing. The resulting tree depicted in the figure is based on an optimized and balanced construction of the tree using a breadth-first search algorithm.

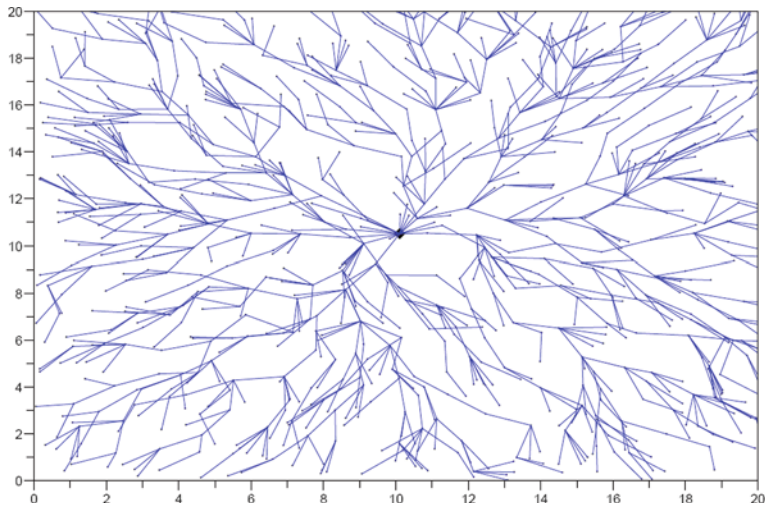


Fig. 15.15 Tree construction after the virtual coordinates assignment [22]

[1] Figure is taken from [22].

In the same way the proximal labeling process presented in [9] uses a depth-first traversal to build a tree. A flood tree is a tree obtained as follows. A root node is selected and it transmits a request message accepting up to a constant number, k, of replies. All the nodes accepted are linked to the root network, and the procedure is repeated recursively until all the nodes in the network are linked to the tree. After the flood tree is built the nodes are labeled in increasing order following a depth-first traversal of tree, leaving gaps between successive labels. Unlike [22] the skipping in the labeling procedure allows room for later insertions of new nodes.

The *LTP* protocol proposed in [8] also uses a tree for routing decision but in such a way that the path from any two nodes in the network is embedded in the label. In LTP, the tree is built iteratively from the root to the leaves. At bootstrap, a node is designed as root. This node may be a special node such as a fixed landmark or a selected node. At each step, every freshly labeled node queries its unlabeled neighbors and then gives a label to each answering node. If $l(u)$ is the label of node u and $|l(u)|$ the size of this label, the kth neighbor of node u is labeled $l(u)k$. Note that a node y already labeled as $l(y)$ may also answer to a node v to get a new label if and only if $|l(y)| > |l(v)| + 1$. The built tree gives the shortest path in the number of hops from the root to any other node. The distance used in the tree is based on label size and common prefix which can give the hop distance between any two nodes of the network. Thus the distance between node a and node b is $d_T(a, b) = ||l(a)| - |l(c)|| + ||l(c)| - |l(b)||$ where c is the lowest common ancestor of a and b and $|l(a)|$ (resp. $|l(b)|$) is the label size of node a (resp. of node b). Figure 15.16 shows an example of the labeling of LTP. However, this is worth noting that though reducing the stretch factor compared to other tree-based georouting protocols, LTP still presents a non-negligible stretch factor. This latter one can be even more reduced by using several trees and letting nodes following the most adequate tree at each routing decision. Indeed, additional trees mean additional bootstrap costs to build them. Nevertheless, studies have shown that using two trees

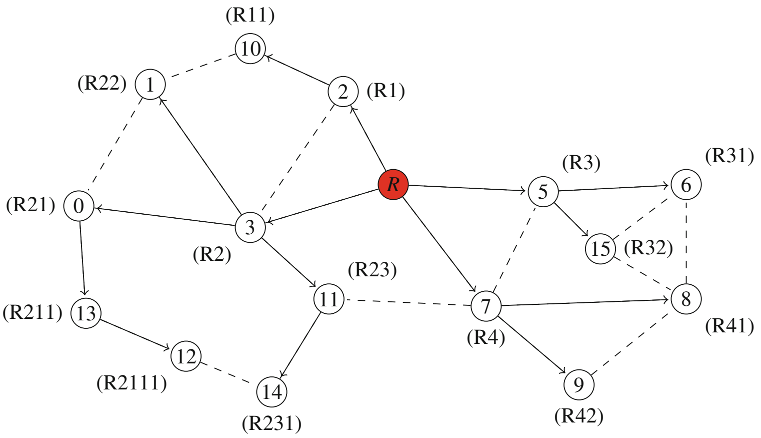


Fig. 15.16 Label results from the LTP protocol [8]

is enough to get a very low stretch factor. The main drawback of LTP (independently of drawbacks relative to tree construction and common to all protocols) is that if the tree is not well balanced, some labels may be very huge and generate a memory overhead.

The tree root is node 4 and has label R. Node 13 is labeled $R211$ since it is the first child of node 0 which has label $R21$. Node 3 has label $R2$ as the second child of the root. From Fig. 15.16 the distance between node 9 and node 5 is thus $d_T(9,5) = ||l(9)| - |l(4)|| + ||l(4)| - |l(5)|| = |3-1| + |1-2| = 3$ hops.

As in every georouting protocol, each node is aware of the labels of itself, of its neighbors, and of the destination. Routing decision is then performed based on all these information and will try as much as possible to avoid to go through the root. The packet is forwarded to the neighbor whose d_T distance to destination is the lowest one. In this way, the routing path may follow "shortcuts" in the tree, decreasing the stretch factor. An example of such a case is given in Fig. 15.16 while considering the routing from node 9 to node 5. Node 7 sends directly to node 5, avoiding the tree root. This allows to reduce the stretch factor.

Algorithms based on tree constructions guarantee packet delivery but generate a much overhead, moreover resulting paths have high stretch factor since routes follow the tree and usually pass through the tree root.

In [37] the stretch factor is even more reduced but at the price of a huge message overhead and construction latency. In this paper, the authors propose *ABVCap*, an axis-based virtual coordinate assignment which is very close to tree-based coordinate system with multiple trees. In ABVCap, a node u is assigned at least one 5-tuple virtual coordinate $(u_{\mathrm{lo}}, u_{\mathrm{la}}, u_{\mathrm{rp}}, u_{\mathrm{up}}, u_{\mathrm{down}})$. According to its relative geographical position and node density (mean node degree), every node will be assigned one or more 5-tuple virtual coordinates (to an infinity). It is worth noting here that every node in the network does not have the same number of 5-tuple coordinate unlike all other proposals. The first two coordinates $(u_{\mathrm{lo}}, u_{\mathrm{la}})$ are used as location information. These locations are longitude and latitude. The three last coordinates $(u_{\mathrm{rp}}, u_{\mathrm{up}}, u_{\mathrm{down}})$ are used for routing. The coordinate assignment is split into four phases. In the first phase, four anchors (X,Y,Z,Z') are selected. In the second phase axes are established: latitude parallel (X–Y) and meridians (Z,Z'). These anchors and axis are fixed for the whole network and the same for every node. Based on these axis, some virtual meridians and parallels are virtually drawn over nodes through flooding from every anchor. Based on this, nodes are assigned their lo, la, and rp coordinates according to their relative distance to nodes that lie on meridian and/or parallel lines through a complex method. In the last phase, up and down coordinates are assigned. Figure 15.17 shows the resulting axis assignment of ABVCap.[2]

The routing process is then performed as follows. The routing packet contains the longitude and the latitude of the destination and a direction bit which is set to 1 if $s_{\mathrm{lo}} < d_{\mathrm{lo}}$ where s is the source and d the destination. Each node knows its multiple

[2] The figure is taken from [37].

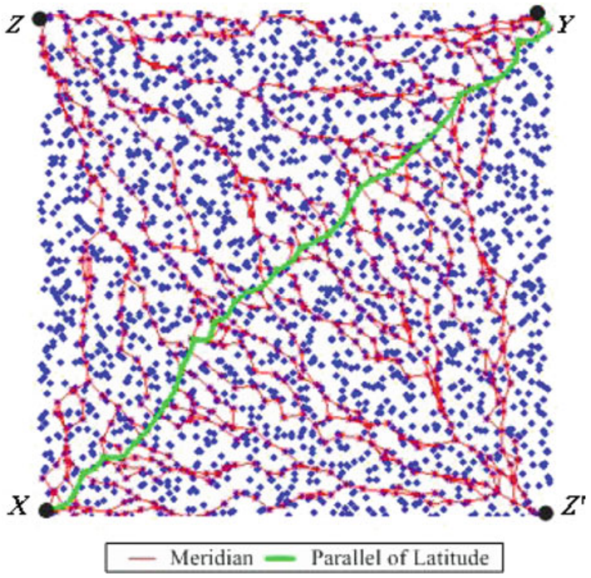


Fig. 15.17 ABVCap virtual coordinates assignment [37]

own coordinates, all multiple coordinates of all of its one-hop neighbors and of the destination. The routing decision is done in two phases. First, the source chooses the 5-tuple coordinates of the destination (among the set of coordinates of the destination) that minimizes the difference $s_{\text{lo}} - d_{\text{lo}}$. Second, the next hop choice is done based on Euclidean distance computed as $\delta_{\text{e}}(u, v) = \sqrt{(u_{\text{lo}} - v_{\text{lo}})^2 + (u_{\text{la}} - v_{\text{la}})^2}$. The authors prove that their routing protocol guarantees delivery and only use greedy forwarding such as in LTP. However, compared to LTP, the stretch factor[3] of ABVCap is lower and it is more robust to changes in the network, but the initialization phase is costly in number of messages and latency.

So far in this section, literature has witnessed enhancements in terms of guaranteed delivery to the detriment of stretch factor (and thus energy consumption and bandwidth overhead). In addition, energy efficiency has been poorly addressed.

15.3.2.3 Energy Efficiency

In this section, we provide an in-depth description of *HECTOR* [24] which is, to the best of our knowledge, the only georouting protocol that both is energy efficient and guarantees packet delivery on top of virtual coordinates. This protocol mixes the use of tree-based coordinate system and landmark-based coordinate system and therefore highlights the previous sections.

[3] The stretch factor is the difference in route length between the one computed by the algorithm and the optimal path.

The routing protocol and virtual coordinate assignment used in [24] take benefits from both tree-based labels like in [8] and landmark-based coordinates like in VCap [7] or VCost [11], with the use of Hamming distance to provide a routing solution based on virtual coordinates with a short initialization phase. The aim of this work was to provide a georouting algorithm with no position information which guarantees delivery and is energy efficient. The authors of [24] propose two algorithms according to the ability of nodes to adjust their range: *HECTOR'* when nodes cannot tune their transmission range and HECTOR when they can. *HECTOR* is thus also energy efficient. Both algorithms use both kinds of coordinates: LTP labels and landmark-based labels.

The routing decision of HECTOR' (resp. HECTOR) is done in such a way that the tree label distance (computes on T labels, noted d_T like for LTP) is always decreasing or at least stationary, and the V coordinates (VCap-like coordinates, Hamming distance on V coordinates is noted d_V) are used to reduce the stretch factor. The basic idea is the following. A source node s holding a packet for a destination node d performs a greedy routing scheme in a VCap (resp. VCost) fashion for HECTOR' (resp. HECTOR). In order to avoid to be trapped in a local minimum, the routing algorithm selects the next hop with regard to not only the V coordinates but also the T labels. The routing process runs as follows. When node u receives a message for node d, it first considers its neighbors in the forward direction, based on both their T and V coordinates. It only considers node v for which d_T distance (distance based on the tree) toward d is equal or smaller than the tree distance between u and d ($d_T(v, d) \leq d_T(u, d)$). We note $N_T(u)$ the set of neighbors of node u such that $d_T(v, d) \leq d_T(u, d)$. Such neighbors always exist ($N_T(u) \neq \emptyset$) if the network is connected because of the tree construction. The node u first checks whether any one of these nodes also provides a progress with respect to V coordinates. We note $N_V(u)$ the set of neighbors of u providing a progress toward the destination with regard to V coordinates, i.e., such that $d_V(v, d) \leq d_V(u, d)$. If $H = N_T(u) \cap \{N_V(u) \cup v\ |d_T(v, d) = d_T(u, d)\}$ is the set of such nodes and $H \neq \emptyset$ then u selects its next hop among the nodes in H thus reducing the distance (resp. optimizing the cost over progress) toward the destination regarding coordinates V and not increasing distance regarding T labels. If $H = \emptyset$, node u selects its neighbor v which provides the best progress to the destination regarding T labels. Such a node always exists since there always exists exactly one path in the tree between any two nodes. In case of ties, the next hop is chosen at random between candidates.

As already mentioned, the progress in HECTOR is computed based on a cost-over-progress fashion. If $H \neq \emptyset$ then u selects its next hop among the nodes in H as the node v which provides the best ratio cost over progress to the destination regarding the virtual coordinates (v such that $\mathrm{COP}_V(u, v, d) = \min_{w \in N_V(u)} \mathrm{COP}_V(w)$). If $H = \emptyset$, v is chosen such that $\mathrm{COP}_T(u, v, d) = \min_{w \in N_T(u)} \mathrm{COP}_T(u, w, d)$. This variant needs the distance between nodes to evaluate the cost of the transmission.

Let us take Figs. 15.11 and 15.16 to illustrate the behavior of HECTOR and HECTOR'. The routing between node 14 (L_3) and node 5 gives an example of the guaranteed delivery provided by HECTOR' and HECTOR. Following VCap or VCost scheme, node 11 is the next hop chosen by node 14. In VCap or VCost, node

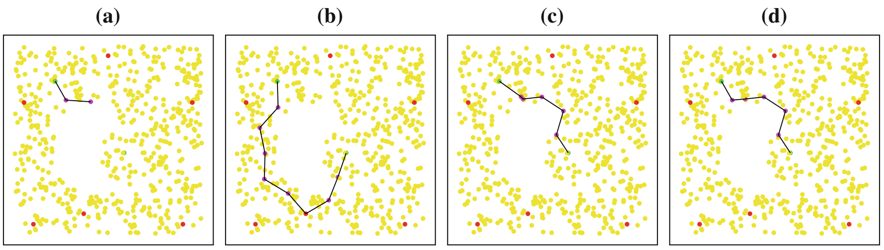


Fig. 15.18 Illustration of the paths followed by each algorithm with the use of five landmarks. VCost/VCap fails after the second hop, and LTP passes through the tree root. HECTOR and HECTOR' combine both T and V coordinates: (**a**) VCost, (**b**) LTP, (**c**) HECTOR', (**d**) HECTOR

11 is a dead end if we use Hamming distance since distance from node 11 to node 5 is equal to 2 and distance from node 7 to node 5 is also 2. As HECTOR' and HECTOR use T labels to avoid reaching a dead end, node 3 is chosen as the next hop of node 11 since $d_T(3, 5) < d_T(11, 5)$. Packet will then go through node 4 as its last relay to node 5.

Figure 15.18 shows the path shapes of different routing algorithms depending on the coordinate system used. This figure shows how HECTOR' and HECTOR take benefit from T labels and V coordinates to provide a guaranteed delivery routing with an enhanced stretch factor.

The following theorems and lemmas prove that (in Lemma 1) at each step of the algorithm there is always a progress at least on T coordinates, that the resulting path is loop free (Lemma 2), and that there is always a next hop that is closer to the destination than the current node (Lemma 3). These lemmas lead to Theorem 3 which shows that the routing process guarantees packet delivery.

Lemma 1 *A packet cannot transit from a node u to another node v if $V(u) = V(v)$ (or if $d_V(u, d) < d_V(v, d)$) unless there is a positive progress regarding T coordinates (if $d_T(u, d) > d_T(d, v)$).*

Proof Let us assume that node u holds a packet for a destination w. Suppose that nodes u and v have the same V coordinates ($V(u) = V(v)$) or that v is farther than node u regarding the V coordinates ($d_V(u, w) < d_V(v, w)$), then by definition node $v \notin N_V(u, w)$. Thus $v \notin H$. The selected next hop is thus part of $H' = \{x|\text{COP}_T(u, x, w) = \min_{i \in N_T(u)} COP_T(u, i, w)\}$, which only contains neighbors of u closer to w than u regarding T labels. Thus, if node v is chosen as the next hop, that means that $v \in H'$ and thus provides a progress regarding T coordinates. Note that in the worst case, i.e., when the progress on T coordinates is minimal, the next hop is either the parent or a child of node u. ■

Lemma 2 *HECTOR is loop free.*

Proof Let us assume that node u_0 is the source of a packet, w its destination, and node u_1 the next hop chosen by node u_0. This means that $d_T(u_0, w) \geq d_T(u_1, w)$ is based on H and H' construction. It suffices to show that the next hop chosen by node u_1 cannot be node u_0.

- CASE (1) Let us first assume that node $u_1 \in N_V(u_0, w)$. This means that $d_V(u_1, w) < d_V(u_0, w)$, thus $u_0 \notin N_V(u_1, w)$. Thus, node u_0 could be selected as the next hop of u_1 if and only if it provides a progress regarding T coordinates $d_T(u_0, w) < d_T(u_1, w)$ (Lemma 1). But according to Lemma 1, if u_1 is the next hop chosen by u_0, we have $d_T(u_1, w) \leq d_T(u_0, w)$. We thus reach a contradiction and thus, the next hop chosen by node u_1 cannot be node u_0.
- CASE (2) Let us now assume that node $u_1 \notin N_V(u_0, w)$, Lemma 1 tells us that u_1 is a parent, a child, or a node that maximizes the cost over progress toward w from node u_0 thus, if u_1 is the next hop for u_0, $d_T(u_0, w) > d_T(u_1, w)$ and node u_0 cannot be selected as the next hop for node u_1.

By transitivity of $d_T()$ and $d_V()$ we cannot have a path $u_0, ..., u_i, ..., u_O$. These two cases imply that the routing protocol is loop free. ■

Lemma 3 *Using HECTOR's coordinate system, there always is a next hop that is closer to the destination regarding virtual coordinates.*

Proof Let us consider a source u and a destination w. By construction, if a node in $N_V(u, w)$ is chosen as the next hop, this ensures a progress in the V coordinates. If the next hop is chosen in $N_T(u, w)$ this ensures a progress in the tree toward the destination. ■

It is worth noting that the progress made on V coordinates is more important than the progress made on T coordinates in the geographical space. Indeed, the next hop in the T labels can have the same V coordinates and thus more or less the same Euclidean distance to the destination. The greedy aspect provided by this algorithm makes it simple, memoryless, and scalable. It is interesting to see here how the simple combination of two coordinates system can enhance the performances (path length, energy efficiency, guaranteed delivery) of georouting protocols. It is worth noting that the authors of [24] mainly focus their work on energy efficiency by the use of cost over progress for next hop selection. As a result, the idea defended by authors can be applied to reduce the hop distance, the Euclidean distance, and any metric for next hop selection. The authors also highlight that V coordinates can be replaced by real geographic routing if real geographic coordinates are available.

Theorem 3 *HECTOR guarantees packet delivery.*

Proof Each node has a unique label due to the labeling process. This ensures that the destination of a packet is unique and that at each step of the routing protocol, a next hop closer to the destination can be found. Based on Lemma 1, Lemma 2, and Lemma 3 if a path exists (if the network is connected), the routing protocol will find one in a greedy way. ■

15.3.2.4 Tree-Based Positioning System Issues

The main drawbacks of tree-based coordinate systems are the building process and tree maintenance. First, the root's choice can strongly impact the tree shape and thus the routing performance in terms of path length. Second, since tree construction is associated with a labeling process and is done in a depth-first search of a breadth-first search process, the label's size may have different sizes for each node. Finally, the tree maintenance is a difficult task since the tree construction is based and initialized by centralized nodes; if a node dies, the whole tree has to be rebuilt.

15.4 Conclusion

Indeed, in this chapter, we have reviewed most of the literature georouting protocols. Table 15.1 sums up the different categories and algorithms, with respect to their characteristics.

Table 15.1 Classification of georouting protocols

	Exact position	Virtual position
Hop based	Greedy [14], MFR [35]	VCap [7], Gliders [13]
Directional	DIR [18], DREAM [1], LAR [17]	
Energy efficient (EE)	COP [19], NFP [16]	VCost [11]
Guaranteed delivery (GD)	GFG [4]	LTP [8], [22], ABVCap [37]
EE+GD	SPFSP [31], EtE [12]	HECTOR [24]

Energy-efficient algorithms assume that nodes are able to compute the Euclidean distance between themselves and their neighbors. Indeed, each algorithm presents its strengths and weaknesses.

Most of the time, there is no more suitable algorithm. Trade-offs have to be made with respect to the context, their environment, and node abilities: geographical or virtual coordinates, abilities to compute an Euclidean distance, computing resources, memory size, etc.

Wireless links are prone to multiple physical phenomena such as interference, collisions, shadowing, fading. They also are impacted by obstacles, buildings, and meteorology conditions. All these features make the transmission unpredictable and unreliable. Thus, next steps of research will be to cope the different algorithm characteristics mentioned in this chapter with the impact of their application in real wireless environments.

References

1. S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A distance routing effect algorithm for mobility (dream). In: *Proceedings of the 4th ACM Annual International. Conference on Mobile Computing and Networking (MOBICOM)*, pages 76–84, Dallas, Texas, 1998.
2. F. Benbadis, J-J. Puig, M.Dias de Amorim, C. Chaudet, T. Friedman, and D. Simplot-Ryl. Jumps: Enhanced hop-count positioning in sensor networks using multiple coordinates. *International Journal on Ad Hoc & Sensor Wireless Networks*, 2008.
3. P. Boone, E. Chavez, L. Gleitzky, E. Kranakis, J. Opartny, G. Salazar and J. Urrutia Morelia Test. Improving the efficiency of the gabriel test and face routing in Ad-hoc Networks *Lecture Notes in Computer Science* , 3104:23–24, 2004, 2008.

4. P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In: *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication (DIAL-M)*, pages 48–55, Seattle, WA, August, 1999.
5. J. Carle and D. Simplot-Ryl. Energy efficient area monitoring by sensor networks. *IEEE Computer Magazine*, 37:40–46, 2004.
6. S. Capkun, M. Hamdi and J-P Hubaux. GPS-Free positioning in mobile ad-hoc networks In: *Proceedings of the Hawaii International Conference on System Sciences HICSS*, Hawaii, USA, 2001, 2005.
7. A. Caruso, S. Chessa, S. De, and A. Urpi. Gps free coordinate assignment and routing in wireless sensor networks. In: *Proceedings of the 24th Conference of the IEEE Communications Society (INFOCOM)*, volume 1, pages 150–160, Miami, USA, March, 2005.
8. E. Chávez, N. Mitton, and H. Tejeda. Routing in wireless networks with position trees. In: *Proceedings of the 6th International Conference on AD-HOC Networks & Wireless (Ad Hoc Now)*, Morelia, Mexico, September 2007.
9. E. Chávez, M. Fraser, and H. Tejeda. Proximal labeling for oblivious routing in wireless ad hoc networks. In *ADHOC-NOW '09: Proceedings of the 8th International Conference on Ad-Hoc, Mobile and Wireless Networks*, Springer, Berlin, pages 360–365, 2009.
10. T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol (OLSR), 2003. RFC 3626.
11. E. H. Elhafsi, N. Mitton, and D. Simplot-Ryl. Cost over progress based energy efficient routing over virtual coordinates in wireless sensor networks. In: *Proceedings of IEEE International Workshop: From Theory to Practice in Wireless Sensor Networks (t2pWSN)*, Helsinki, Finland, 2007.
12. E.H. Elhafsi, N. Mitton, and D. Simplot-Ryl. End-to-end energy efficient geographic path discovery with guaranteed delivery in ad hoc and sensor networks. In: *Proceedings of the 19th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Cannes, France, September 2008.
13. Q. Fang, J. Gao, L.J. Guibas, V. de Silva, and Li Zhang. GLIDER: gradient landmark-based distributed routing for sensor networks. In: *Proceedings of the 24th Conference of the IEEE Communications Society (INFOCOM)*, volume 1, pages 339–350, Miami, USA, March 2005.
14. G.G. Finn. Routing and addressing problems in large metropolitan-scale. *Internetworks, ISI Research Report ISU/RR-87-180*, March 1987.
15. H. Frey, and I. Stojmenovic. On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks. *Proceedings of the 12th ACM Annual International Conference on Mobile Computing and Networking (MOBICOM)*, Los Angeles, CA, ages 390–401, September 2006.
16. T.-C. Hou and V. Li. Transmission range control in multihop packet radio networks. *Communications, IEEE Transactions on [legacy, pre - 1988]*, 34(1):38–44, 1986.
17. Y-B Ko and N. H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. In: *Proceedings of the 4th ACM Annual Int. Conference on Mobile Computing and Networking (MOBICOM)*, pages 66–75, Dallas, Texas, 1998.
18. E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In: *Proceedings of the 11th Canadian Conference on Computational Geometry (CCCG)*, Vancouver, Canda, 1999.
19. J. Kuruvila, A. Nayak, and I. Stojmenovic. Progress and location based localized power aware routing for ah hoc sensor wireless networks. *IJDSN*, 2:147–159, 2006.
20. J. Li, L. Gewali, H. Selvaraj and V. Muthukumar. Hybrid Greedy/Face Routing for Ad-Hoc Sensor Networks. In *DSD Journal*, Los Alamitos, CA, 0:574–578, 2004.
21. X. Lin and I. Stojmenovic. Geographic distance routing inad hoc wireless networks. Technical Report TR-98-10, SITE, University of Ottawa, December 1998.
22. K. Liu and Nael Abu-Ghazaleh. Stateless and guaranteed geometric routing on virtual coordinate systems. In: *Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 340–346, October 2006.

23. Ke Liu and Nael Abu-Ghazaleh. Aligned virtual coordinates for greedy routing in wsns. In *Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 377–386, October 2006.
24. N. Mitton, T. Razafindralambo, D. Simplot-Ryl, and I. Stojmenovic. Hector is an energy efficient tree-based optimized routing protocol for wireless networks. In *Proceedings of the 4th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, Wuhan, China, December 2008.
25. N. Mitton, D. Simplot-Ryl, and I. Stojmenovic. Guaranteed delivery for geographical any-casting in wireless multi-sink sensor and sensor-actor networks. In: *Proceedings of the 28th Conference on Computer Communications (INFOCOM)*, Rio de Janeiro, Brasil, April 2009.
26. Nelson, R. and Kleinrock L. The spatial capacity of a slotted ALOHA multihop packet radio network with capture In *IEEE Transactions on Communications*, 32, 6: 684–689, June 1984.
27. D. Niculescu and B. Nath. Ad hoc positioning system (APS). In: *Proceedings of IEEE Global communications conference GLOBECOM*, San Antonio, USA, 2001.
28. N. Patwari, J.N. Ash, S. Kyperountas, A.O. III Hero, R.L. Moses, and N.S. Correal. Locating the nodes: Cooperative localization in wireless sensor networks. *Signal Processing Magazine, IEEE*, 22(4):54–69, July 2005.
29. C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector routing, 2003. RFC 3561.
30. A. Rao, C. H. Papadimitriou, S. Shenker and I. Stoica. Geographic routing without location information, *Proceedings of the 9th ACM Annual International Conference on Mobile Computing and Networking* (*MOBICOM*), 96–108, 2003.
31. J.A. Sanchez and P.M. Ruiz. Exploiting local knowledge to enhance energy-efficient geographic routing. In: *Proceedings of the 2nd International Conference on Mobile Ad-hoc and Sensor Networks MSN*, pages 567–578, December 2006.
32. I. Stojmenovic, S. Datta. Power and cost Aware Localized Routing with guaranteed delivery in wireless networks. *Wireless Communication and Mobile Computing*, 2(4):175–188, 2004.
33. I. Stojmenovic and X. Lin. Loop-Free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks *IEEE Transactions Parallel Distribution Systems (TPDS)*, 12(10):1023–1032, 2001.
34. I. Stojmenovic and X. Lin. Power-aware localized routing in wireless networks. *IEEE Transactions Parallel Distribution Systems (TPDS)*, 12(11):1122–1133, 2001.
35. H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE transaction on communications*, com-22(3):246–257, 1984.
36. G. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition Journal*, 12(4):261–268, 1980.
37. M-J Tsai, H-Y Yang, and W-Q Huang. Axis-based virtual coordinate assignment protocol and delivery-guaranteed routing protocol in wireless sensor networks. In: *Proceedings of the 26rd Conference of the IEEE Communications Society (INFOCOM)*, volume 1, pages 2234–2242, May 2007.
38. G. Voronoi. New applications, from continuous parameter to quadratic shape theory. *Journal für die Reine und Angewandte Mathematik*, 133:97–178, 1907.
39. Y. Wang, W.-Z. Song, W. Wang, X.-Y. Li, T.A. Dahlberg. LEARN: Localized energy aware restricted neighborhood routing for Ad Hoc networks. In: *Proceedings of the 3rd Annual IEEE Sensor and Ad Hoc Communications and Networks(SECON)*, 2, 508–517, 2006.
40. T. Watteyne, I. Augé-Blum, M. Dohler, S. Ubeda, and D. Barthel, Centroid virtual coordinates - A novel near-shortest path routing paradigm. To appera in *International Journal of Computer and Telecommunications Networking, Elsevier*, 53(10):1697–1711, to appear.

Part V
Energy Optimization

Chapter 16
Energy-Balanced Data Propagation in Wireless Sensor Networks

Pierre Leone, Sotiris Nikoletseas, and José D.P. Rolim

Abstract We study the problem of energy-balanced data propagation in wireless sensor networks. The energy balance property guarantees that the average per sensor energy dissipation is the same for all sensors in the network, during the entire execution of the data propagation protocol. This property is important since it prolongs the network's lifetime by avoiding early energy depletion of sensors. We first present a basic algorithm that in each step probabilistically decides whether to propagate data one-hop towards the final destination (the sink), or to send data directly to the sink. This randomized choice balances the (cheap, but slow) one-hop transmissions with the direct transmissions to the sink, which are more expensive but "bypass" the bottleneck region around the sink and propagate data fast. Note that in most protocols, the sensors lying closer to the sink tend to be overused and "die out" early. By a detailed analysis we precisely estimate the probabilities for each propagation choice in order to guarantee energy balance. The needed estimations can easily be performed by current technology sensors using simple to obtain information. Under some assumptions, we also derive a closed form for these probabilities. The fact (shown by our analysis) that direct (expensive) transmissions to the sink are needed only rarely, shows that our protocol, besides energy balanced, is also energy efficient. We then enhance this basic result with some recent findings including a generalized algorithm and demonstrating the optimality of this two-way probabilistic data propagation, as well as providing formal proofs of the energy optimality of the energy balance property.

16.1 Introduction

Wireless sensor networks are usually comprised of a very large number of low cost micro-sensors, which distributively form an ad hoc network by locally communicating with each other wirelessly. The micro-sensors are deployed in an area of interest,

P. Leone (✉)
University of Geneva, Geneva, Switzerland
e-mail: pierre.leone@unige.ch

in order to, e.g. monitor crucial events and propagate the collected results to a base station, which is usually called the *sink*. There are many important applications of such networks ranging from security applications to environmental monitoring, based on the existence of a wide variety of sensor types, such as for temperature, motion, noise, seismic activity (see [1, 2] for excellent surveys).

The recent dramatic progress in micro-electro-mechanical systems (MEMS) regarding the design and implementation of low-power electronics and low-power RF and laser communication systems has already led to small sized, low power and relatively inexpensive wireless sensors. However, wireless micro-sensors have to operate under severe limitations of their computational power, data storage, the quality of communication and, most crucially, their available amount of energy (battery). Thus, the efficient distributed cooperation of sensor devices towards achieving a large computational and communication goal is a challenging task.

An important goal in the design and efficient implementation of wireless sensor networks is to save energy and keep the network functional for as long as possible. Towards this goal, various approaches including hop-by-hop transmission techniques [4, 12, 13], as well as clustering techniques [11] and alternating power saving modes [21] have been proposed. Energy aware data gathering protocols have been introduced in [3].

All such techniques do not *explicitly* take care of possible overuse of certain sensors in the network. As an example, we remark that in hop-by-hop transmissions towards the sink, the sensors lying closer to the sink tend to be utilized exhaustively (since all data pass through them). Thus, these sensors may die out very early, thus resulting to network collapse although there may be still significant amounts of energy in the other sensors of the network. Also, distant transmissions, e.g. by cluster heads, tend to overuse the cluster heads; in large networks of high data traffic the rotation of cluster heads may not suffice for preventing their energy depletion.

16.1.1 The Main Idea

In the light of the above, one should try to combine energy efficiency with a *balanced spread* of energy consumption among sensors.

In this chapter, we first present and analyze *an energy-efficient and energy-balanced protocol (EBP)* for data propagation in sensor networks. The energy balance property of the algorithm ensures that the average energy dissipation per sensor is the same throughout the execution of the protocol in all sensors.

In particular, for the analysis we assume a random uniform placement of sensors in the network area and a random uniform generation of events. Once a data message reaches a sensor, it is either propagated one-hop closer to the sink or it is sent directly to the sink. The choice of whether to forward one-hop or send directly is done locally at each sensor in a probabilistic manner.

By a detailed analysis, we accurately estimate these probabilities in a recurrent way that can be supported by current sensors' technology. We note that these probabilities are found to be dependent only on the network size and the distance of each particular sensor from the sink. Under certain assumptions, we also provide a simple closed form for the probability p_i to forward data one-hop further. In particular, we show that $p_i = 1 - \frac{3x}{(i+1)(i-1)}$, where x is a free (e.g. controlling) parameter and i is related to the sensor's distance to the sink.

We note that although we here present a specific model for energy dissipation in data transmission, our results can be easily extended to any energy cost model.

We also note that this basic protocol, besides being energy balanced is also energy efficient, in the sense that our analysis shows that for guaranteeing energy balancedness, most transmissions must be one-hop transmissions; thus data are mostly transmitted along small distances and thus the overall energy consumption in the network is kept low. We then enhance this basic result with some recent findings demonstrating the optimality of this two-way probabilistic data propagation, as well as formal proofs of the energy optimality of the energy balance property.

Our work has been inspired by the important relevant work of [22] where the authors define the energy balancedness property and propose, analyse and evaluate an energy-optimal and energy-balanced algorithm for sorting in wireless sensor networks. In particular, they consider a single-hop sensor network. Thus, our work extends their approach to the general case of a multi-hop network and for the (quite general) problem of propagating data to the sink.

16.1.2 Roadmap

In the next section we present the network model and define the problem of energy-balanced data propagation. In Sect. 16.3 we provide the basic distributed algorithm. After some preliminary material in Sect. 16.4, in Sect. 16.5 we carry out a detailed analysis of the general case leading to an accurate estimation of data forwarding probabilities and discuss its behaviour for when we vary some parameters of the network. In Sect. 16.6 we provide a simple closed form for these probabilities under some assumptions that we discuss. Finally, in Sect. 16.7 we present a generalized energy balance algorithm, while in Sect. 16.8 we further elaborate on the energy optimality of energy balance algorithms.

16.2 The Model and the Problem

We assume that crucial events, that should be reported to a control centre, occur in the network area. Furthermore, we assume that these events are happening at random uniform positions. Let N be their total number in a certain period (i.e. during the execution of the protocol).

The sensors are spread in the network area randomly uniformly so their number in a certain area is proportional to the area's size. There is a single point in the network, which we call the "sink" S, that represents the fixed control centre where data about event realization should be reported to. The sink is very powerful, in terms of both energy and computing power. Sensors can be aware of the direction (and position) of the sink, as well as of their distance to the sink. Such information can be easily obtained during a set-up phase by having the (powerful) sink broadcast low overhead control messages to the entire network area. We assume that the transmission range of sensors can vary with time (in fact for each sensor our protocol may use only two different ranges: R and $i \cdot R$, where R is set by the network operator and i can be defined as $\lceil d/R \rceil$ with d being the sensor's distance from the sink). We assume a strong network model where all sensors are able to reach the sink in one-hop. Similar modelling assumptions are made in the state of the art (see, e.g. the LEACH clustering protocol [11]). However, we note that this strong assumption does not trivialize the problem: former protocols with the same strong modelling assumptions are not energy balanced (see Sect. 16.2). We do not address what is the largest allowable transmission range (e.g. the largest i above); this would be limited by technology and interferences. However, taking a sufficiently large angle ϕ and/or by hierarchically taking multiple sets of sectors, we can cover the whole area without increasing i beyond the maximum allowable limit. The sensors in our model do not move.

We virtually "cover" the network area by a disk sector of angle ϕ (see Fig. 16.1). The disk sector is divided into n ring sectors or "slices". The first slice has radius R (i.e. the sensors' transmission range). Slice i $(2 \leq i \leq n)$ is defined by two successive disk sectors, one of radius $i \cdot R$ and the other of radius $(i-1) \cdot R$. Taking a sufficiently large angle ϕ and/or by taking multiple sectors in a hierarchical way, we can cover the whole network area.

As far as energy dissipation is concerned, we assume that the energy spent at a sensor when transmitting data messages is proportional to the square of the transmitting distance. Our protocol's performance analysis can be however extended to any energy cost model. The size of messages is considered to be constant. We note that the energy dissipation for receiving a data message is not always negligible with respect to the energy when sending such as in case when transmissions are too short and/or radio electronics energy is high (see [11]). In the analysis, initially, we only count (for simplicity) energy spent during transmissions. Since however in our protocol (see next section) there is one receipt for each transmission, it is

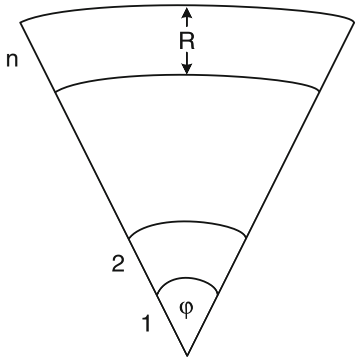


Fig. 16.1 Sensor network with n ring sectors, angle ϕ and ring "width" R

clear that even when energy during receipt is more or less the same as energy during transmissions, the analysis can be extended easily to the full case (counting both transmissions and receipts).

Definition 1 The area between two consecutive cycle sectors is called a ring sector (or "slice"). Let T_i $(1 \leq i \leq n)$ be the ith ring sector of the network. T_1 stands for the ring sector with centre the sink and radius equal to R.

Definition 2 Let S_i be the area size of the ring sector T_i of the network $(1 \leq i \leq n)$.

We wish to solve the "*energy-balanced data propagation problem*", i.e. to propagate data to the sink in such a way that the "average"energy dissipation in each sensor is at each time the same. The average energy dissipation per sensor is taken to be the fraction of the total energy spent by sensors in a ring sector over the number of sensors in that sector. Because of our assumption that the number of sensors in an area is proportional to the area size, the average energy dissipation per sensor is calculated by dividing the total energy spent in a sector by the sector size.

We do not consider medium access aspects, assuming the existence of an ideal underlying MAC protocol.

16.3 The EBP Distributed Data Propagation Protocol

We assume that each event is sensed by only one sensor. This assumption is not restrictive since we may consider multiple sensing and propagation of an event by various sensors as sensing and propagation of many different events, assuming methods to resolve symmetry (such as clustering, leader election) in cases like when sensors are acoustic and the event is a thunderclap. A sensor sensing an event generates a data message which should be eventually delivered to the sink. On each ring sector, T_i, a number of events occur and a corresponding number of messages (one for each event) are generated.

Randomization is used to achieve some "load balancing" by evenly spreading the "load" (energy dissipation). In particular, on ring sector T_i each event is propagated to T_{i-1} (i.e. the "*next*" sector towards the sink) with probability p_i, while with probability $1 - p_i$ it is propagated directly to the sink S. Each message in T_i is handled stochastically independently of the other events' messages.

The choice of probability p_i for T_i is made so as the average energy consumption per area unit (and thus per sensor) is the same for the whole network. There is a trade-off from choosing p_i: if p_i increases then transmissions tend to happen locally, thus energy consumption is low, however sensors closer to the sink tend to be overused since all data pass through them. On the other hand, if p_i decreases, there are distant transmissions (thus a lot of energy is consumed) however propagation is faster and closer to sink particles are bypassed. Calculating the appropriate probability p_i for each T_i and solving the problem of energy balance are very important since it combines efficient data propagation with increased network's lifetime.

By using an underlying subprotocol (such as [4, 12]) we can guarantee that only one "next hop" sensor receives the transmitted message. Such a subprotocol can be any protocol that uses a (cheap) search phase to locate a "next" particle towards the sink and then directly transmits to that particle only. Note also that data messages are of fixed size, i.e. no further information is added to a message on its route towards the sink.

Our protocol is (a) *distributed*, since each sensor chooses the propagation probability independently of other sensors, (b) it uses only *local* information, in the sense that p_i depends only on i, i.e. a parameter related to the distance from the sink. Note that the distance from the sink information for each sensor can be easily obtained, i.e. during a set-up phase when the sink broadcasts lightweight control messages to the network. Several techniques (including signal attenuation evaluation) can be used to estimate each sensor's distance from the sink. (c) The protocol is *simple*, since it just uses a random choice based only on parameter i.

16.4 Basic Definitions–Preliminaries

We aim at calculating probability p_i for each i in order to ensure the energy balance property.

Lemma 1 The area size, S_1, of the ring sector T_1 is $S_1 = \frac{\phi}{2} \cdot R^2$

Proof From geometry, see also [9]. □

Lemma 2 The relation between the area size of the ring sector T_i and that of T_1 is $S_i = (2i - 1) \cdot S_1$

Proof By using Lemma 1, see also [9]. □

Definition 3 Let λ_i be the probability that an event will occur on the ring sector T_i.

There are n ring sectors in the network.

Lemma 3 Assuming a random uniform generation of events in the network area, the probability λ_i of an event occurring on the ring sector T_i $(1 \le i \le n)$ is

$$\lambda_i = \frac{(2i-1)}{n^2}$$

Proof Firstly, we will compute the probability λ_1. Because of the fact that events occur randomly uniformly in the network area we have that

$$\lambda_1 = \frac{S_1}{\sum_{i=1}^{n} S_i} = \frac{S_1}{\sum_{i=1}^{n}(2i-1)S_1} = \frac{1}{\sum_{i=1}^{n}(2i-1)}$$

$$\lambda_1 = \frac{1}{n^2}$$

Taking into consideration the random uniform distribution of the events in the network area and using Lemma 2, we have that $\lambda_i = (2i-1)\lambda_1$ and the result is proved. □

Let us now consider sector T_i.

Definition 4 An area T_i *"handles"* an event generated in ring sector j if either the message was generated in the area T_i (i.e. $j = i$) or the message was propagated to T_i from the ring sector T_{i+1}.

Definition 5 Let h_i be the number of the messages that are "handled" by the area T_i.

We now define energy ε_{ij} spent for message j when sector i handles it.

Definition 6 Let ε_{ij} be a random variable which measures the energy that dissipates the sector T_i so as to handle the message j. For ε_{ij} we have that

$$\varepsilon_{ij} = \begin{cases} cR^2 & \text{with probability } p_i \\ c(iR)^2 & \text{with probability } 1 - p_i \end{cases}$$

where cR^2 is the energy dissipation for sending a message j from T_i to its adjacent ring sector T_{i-1} and c is a constant.

Thus, the expected energy dissipation in sector i for handling a message is

$$E[\varepsilon_{i,j}] = cR^2 \cdot \left[i^2 - p_i(i^2 - 1)\right] \tag{16.1}$$

Note: The expected energy above is the same for all messages; we use j just for counting purposes.

Definition 7 Let $\mathcal{E}_i$ be the *total energy* spent by sensors in T_i. Clearly

$$\mathcal{E}_i = \sum_{j=1}^{h_i} \varepsilon_{ij} \tag{16.2}$$

Energy balance is defined as follows:

Definition 8 (energy balance) The network is energy balanced if the average per sensor energy dissipation is the same for all sectors, i.e. when

$$\frac{E[\mathcal{E}_i]}{S_i} = \frac{E[\mathcal{E}_j]}{S_j} \quad i, j = 1, \ldots, n \tag{16.3}$$

16.5 The General Solution

We next provide a lemma useful in the estimation of the total energy dissipation in a sector.

Lemma 4 The expected total energy dissipation in sector i is

$$E[\mathcal{E}_i] = E[h_i] \cdot E[\varepsilon_{ik}]$$

Proof

$$\begin{aligned} E[\mathcal{E}_i] &= E\left[\sum_{k=1}^{h_i} \varepsilon_{ik}\right] \\ &= \sum_{n=0}^{N} E\left[\sum_{k=0}^{h_i} (\varepsilon_{ik} \cap h_i = n)\right] \\ &= \sum_{n=0}^{N} E\left[\sum_{k=1}^{h_i} \varepsilon_{ik} \,|h_i = n\right] \cdot \mathbf{P}\{h_i = n\} \end{aligned}$$

Furthermore,

$$\begin{aligned} E\left[\sum_{k=1}^{h_i} \varepsilon_{ik} \,|h_i = n\right] &= E\left[\sum_{k=1}^{n} \varepsilon_{ik}\right] = \sum_{k=1}^{n} E[\varepsilon_{ik}] \\ &= n \cdot E[\varepsilon_{ik}] \end{aligned}$$

Thus, we have from the above that

$$\begin{aligned} E[\mathcal{E}_i] &= \sum_{n=0}^{N} E\left[\sum_{k=1}^{h_i} \varepsilon_{ik} \,|h_i = n\right] \cdot \mathbf{P}\{h_i = n\} \\ &= E[\varepsilon_{ik}] \cdot E[h_i] \end{aligned}$$

Definition 9 Let g_i be the number of the messages that are *generated* in the area T_i. Note that messages are generated in an area only when events occur in this area.

Definition 10 Let f_i be the number of the messages that are *forwarded to* the area T_i.

We note that messages are forwarded to a ring sector (say i) only because of an event generation at a sector $j > i$ and successive one-hop propagations from sector j to sector i.
We notice the following important relation:

$$h_i = g_i + f_i \tag{16.4}$$

which means that the number of messages that area T_i handles equals the number of the messages that are generated in T_i, plus the number of messages that are forwarded to it. Because of event generation according to a probability distribution and also because of the probabilistic nature of message propagation, all three quantities above are random variables. By linearity of expectation, we get

Lemma 5 $E[h_i] = E[g_i] + E[f_i]$

We establish a relationship between $E[f_i]$ and $E[h_{i+1}]$.

Lemma 6 $E[f_i] = p_{i+1} \cdot E[h_{i+1}]$

Proof Let $\delta_{i,j}$ be an indicator random variable that is equal to 1 if area T_i forwards the message j to the area T_{i-1} and 0 otherwise. Thus

$$\delta_{i,j} = \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{with probability } 1 - p_i \end{cases}$$

Clearly, $\delta_{i,j}$ depends only on i, but we add j for counting purposes. Obviously, $E[\delta_{i,j}] = p_i$. It is

$$f_i = \sum_{j=0}^{h_{i+1}} \delta_{i+1,j}$$

Similarly to the proof of Lemma 4, we get

$$E[f_i] = \sum_{n=0}^{N} E\left[\sum_{j=0}^{h_{i+1}} \delta_{i+1,j} \,|h_{i+1} = n\right] \cdot \mathbf{P}\{h_{i+1} = n\}$$

and the proof is completed. □

Recall that according to Definition 8, to achieve the same on the average energy dissipation *per area unit* (and thus per sensor) in the network area, the following equality should hold:

$$E\left[\frac{\sum_{k=1}^{h_i} \varepsilon_{ik}}{S_i}\right] = E\left[\frac{\sum_{k=1}^{h_j} \varepsilon_{jk}}{S_j}\right] \quad \forall i, j \in \{1, \ldots n\} \tag{16.5}$$

i.e. the average energy consumption per sensor should be equal in any two ring sectors. By induction, it suffices to guarantee this for any two adjacent sectors. In what follows, we guarantee the above balance property, requiring a certain recurrence relation to hold. This recurrence basically relates three successive terms of the $E[f_i]$ sequence (the $E[g_i]$ terms depend only on i and on input parameters).

Theorem 1 *To achieve energy balance in the network, the following recurrence equation should hold:*

$$\begin{aligned} a_{i+1}E[f_{i+1}] - (d_i + a_i)E[f_i] + d_{i-1}E[f_{i-1}] = \\ = a_i E[g_i] - a_{i+1}E[g_{i+1}] \end{aligned}$$

where

$$a_i = \frac{i^2}{2i - 1} \qquad d_i = \frac{(i + 1)^2 - 1}{2i + 1}$$

Proof For the case $j = i + 1$ of Eq. (16.5) and using Lemmas 2 and 4 we have

$$\frac{E[h_i]E[\varepsilon_{i,j}]}{S_i} = \frac{E[h_{i+1}]E[\varepsilon_{i+1,j}]}{S_{i+1}} \quad \Leftrightarrow$$

$$\frac{E[h_i]\left[i^2 - p_i(i^2-1)\right]}{(2i-1)} = \frac{E[h_{i+1}]\left\{(i+1)^2 - p_{i+1}[(i+1)^2-1]\right\}}{(2i+1)} \quad \Leftrightarrow$$

$$\tfrac{i^2}{2i-1}E[h_i] - p_iE[h_i]\frac{i^2-1}{2i-1} =$$

$$= \frac{(i+1)^2}{2i+1}E[h_{i+1}] - p_{i+1}E[h_{i+1}]\frac{(i+1)^2-1}{2i+1}$$

Let a_i, d_i be defined as in the theorem statement above. By Lemma 6 we know that $p_iE[h_i] = E[f_{i-1}]$ and by Lemma 5 it is $E[h_i] = E[g_i] + E[f_i]$; thus the last equation becomes

$$a_{i+1}E[f_{i+1}] - (d_i + a_i)E[f_i] + d_{i-1}E[f_{i-1}] =$$

$$= a_iE[g_i] - a_{i+1}E[g_{i+1}]$$

To solve the above recurrence we must compute $E[g_i]$.

Lemma 7 If N is the total number of events that are generated in the network, the mean value of g_i is given by the following relationship:

$$E[g_i] = N \cdot \lambda_i$$

Proof Because the position of each event is independent of other events and because for each sector i, probability λ_i is the same; clearly g_i is binomial with parameters $N,\ \lambda_i$. □

In order to have a simpler recurrence involving only two (successive in fact) terms of the $E[f_i]$ sequence, we will transform the recurrence relation of Theorem 1. It is

$$a_{i+1}E[f_{i+1}] - (d_i + a_i)E[f_i] + d_{i-1}E[f_{i-1}] =$$

$$= a_iE[g_i] - a_{i+1}E[g_{i+1}] \Leftrightarrow$$

$$(a_{i+1}E[f_{i+1}] - d_iE[f_i]) - (a_iE[f_i] - d_{i-1}E[f_{i-1}])$$
$$= a_iE[g_i] - a_{i+1}E[g_{i+1}]$$

Definition 11 Let $t_i = a_{i+1}E[g_{i+1}] - d_iE[g_i]$, for $i \in \{0, 1, \ldots, n-1\}$.

The recurrence relation of Lemma 1 can be expressed as

$$\begin{aligned}&t_i - t_{i-1} = a_i \cdot E[f_i] - a_{i+1} \cdot E[f_{i+1}]\\ &\text{for} \quad i = 1, \dots n-1\\ &\text{with} \quad t_0 = a_1 \cdot E[f_1]\end{aligned}$$

Note that although we do not know yet the value of $E[f_1]$, we can compute the values of t_i as a function of $E[f_1]$. Furthermore, as we prove in the next lemma, this function is linear in $E[f_1]$.

Lemma 8 The recurrence relation

$$\begin{aligned}&t_i - t_{i-1} = a_i \cdot E[f_i] - a_{i+1} \cdot E[f_{i+1}]\\ &\text{for} \quad i = 1, \dots n-1\\ &\text{with} \quad t_0 = a_1 \cdot E[f_1]\end{aligned}$$

has as a solution the function

$$t_i = \sum_{j=1}^{i} \left(a_j E[g_j] - a_{j+1} E[g_{j+1}]\right) + a_1 \cdot E[f_1]$$

Proof The proof is done by induction on i. For $i = 0$, it is obviously true. Let it be true for $i - 1$. For i we have

$$t_i = t_{i-1} + a_i \cdot E[g_i] - a_{i+1} \cdot E[g_{i+1}]$$

By the induction hypothesis we get the solution

$$t_i = \sum_{j=1}^{i} \left(a_j E[g_j] - a_{j+1} E[g_{j+1}]\right) + a_1 \cdot E[f_1] \qquad \square$$

Now the recurrence relation of Theorem 1 is simplified as

$$a_{i+1} \cdot E[f_{i+1}] - d_i \cdot E[f_i] = t_i \quad i = 1, \dots, n-1 \qquad \square$$

Thus, we get a recurrence for sequence $E[f_i]$ involving only two successive terms of the sequence

Theorem 2 *The recurrence relation*

$$a_{i+1} E[f_{i+1}] - d_i E[f_i] = t_i \quad i = 1, \dots n-1$$

where t_i is defined in Lemma 8 has the following solution:

$$E[f_{n-i}] = -\sum_{k=1}^{i} \frac{\prod_{j=k}^{i-1} a_{n-j}}{\prod_{j=k}^{i} d_{n-j}} \cdot t_{n-k}$$

Proof To get an intuitive feeling of the solution, we first present a small case, i.e. $n = 4$. The recurrence relation can be written as a linear system of three variables, $E[f_1]$, $E[f_2]$, $E[f_3]$, and three equations:

$$\begin{aligned} a_2 E[f_2] - d_1 E[f_1] = t_1 \\ a_3 E[f_3] - d_2 E[f_2] = t_2 \\ a_4 E[f_4] - d_3 E[f_3] = t_3 \end{aligned}$$

We note that $E[f_4] = 0$ because no messages are forwarded to the outmost sector. Thus we have

$$E[f_3] = -\frac{t_3}{d_3}$$

$$E[f_2] = -\frac{a_3 t_3}{d_3 d_2} - \frac{t_2}{d_2}$$

$$E[f_1] = -\frac{a_3 a_2 t_3}{d_3 d_2 d_1} - \frac{a_2 t_2}{d_2 d_1} - \frac{t_1}{d_1}$$

We can express t_i as a linear function of $E[f_1]$; thus, the last equation depends only on $E[f_1]$ while the others on $E[f_i]$ and $E[f_1]$. The last of the three equations can give the $E[f_1]$ and then we can calculate the rest $E[f_i]$.

We now proceed to the general n. The proof is done by induction on i. For $i = 1$, we compute the value of $E[f_{n-1}]$. From the recurrence relation we have for $E[f_{n-1}]$

$$a_n E[f_n] - d_{n-1} E[f_{n-1}] = t_{n-1}$$

But, $E[f_n]$ is the number of events that are propagated to ring sector n, which is zero. Also notice that $d_i = 0 \Leftrightarrow i = 0$. Thus

$$E[f_{n-1}] = -\frac{t_{n-1}}{d_{n-1}}$$

which is equal to the solution. So we are done with the base of the induction.

Assuming that the induction hypothesis is correct for $i - 1$, we will prove that it is also correct for i. From the recurrence relation we have that

$$E[f_{n-i}] = \frac{a_{n-i+1}}{d_{n-i}} \cdot E[f_{n-i+1}] - \frac{t_{n-i}}{d_{n-i}}$$

We can safely divide with d_i because the recurrence is defined for $i = 1, \dots, n-1$ and no value of these makes d_i zero. But, from the induction hypothesis we know that

$$E[f_{n-i+1}] = -\sum_{k=1}^{i-1} \frac{\prod_{j=k}^{i-2} a_{n-j}}{\prod_{j=k}^{i-1} d_{n-j}} \cdot t_{n-k}$$

Replacing $E[f_{n-i+1}]$ in the first relation we have

$$E[f_{n-i}] = \frac{a_{n-i+1}}{d_{n-i}} \cdot \left(-\sum_{k=1}^{i-1} \frac{\prod_{j=k}^{i-2} a_{n-j}}{\prod_{j=k}^{i-1} d_{n-j}} \cdot t_{n-k} \right) - \frac{t_{n-i}}{d_{n-i}}$$

We finally get

$$E[f_{n-i}] = -\sum_{k=1}^{i} \frac{\prod_{j=k}^{i-1} a_{n-j}}{\prod_{j=k}^{i} d_{n-j}} \cdot t_{n-k}$$

$$\text{where} \quad \prod_{i}^{i-1} a_i = 1$$

The full expression for $E[f_i]$ can be found by substituting i with $n-i$, thus

$$E[f_i] = -\sum_{k=1}^{n-i} \frac{\prod_{j=k}^{n-i+1} a_{n-j}}{\prod_{j=k}^{n-i} d_{n-j}} \cdot$$

$$\cdot \left(\sum_{j=1}^{n-k} (a_j E[g_j] - a_{j+1} E[g_{j+1}]) + a_1 \cdot E[f_1] \right)$$

where $\prod_{i}^{i-1} a_i = 1$.

We note that all the parameters of the recurrence solution above are expressed as a function of $E[f_1]$ and i. So as to compute them, we first compute the value of $E[f_1]$. Then we can compute all the other parameters by replacing the already computed $E[f_1]$. □

Now, the calculation of the probabilities p_i is quite easy.

Theorem 3 *The energy balance property is achieved if any ring sector (say T_i) propagates each message it handles with probability p_i to the next ring sector, T_{i-1}, and with probability $1 - p_i$ it propagates the message directly to the sink. The value of each p_i is given by the following relation:*

$$p_i = \frac{E[f_{i-1}]}{E[g_i] + E[f_i]}$$

where the values of $E[f_i]$ and $E[g_i]$ are obtained from Theorem 2 and Lemma 7, respectively.

Proof From Lemma 6 we know that $E[f_{i-1}] = p_i E[h_i]$ and also by Lemma 5 we know that $E[h_i] = E[g_i] + E[f_i]$. □ ⊠

Remark 1 Note that interestingly, p_i's are *independent* of the number N of the events that occur in the network, since p_i's depend only on i and on the number of ring sectors n (which is broadcast to sectors by the sink). Thus the protocol assumes only local information.

We note that the analysis above allows the exact derivation of probabilities p_i's as a function of i and n which (although complicated and not obviously leading to a closed form) can be easily calculated by the sensors in the network by carrying out very simple calculations.

We conclude with a correctness proof of the data propagation protocol.

Theorem 4 Given that the energy is each time on the average the same in all network sensors, each message will finally get to the sink.

Proof Consider a message that is generated in a certain ring sector, say T_i. Because of the energy balance property, all sensors are operational; thus, in each step of the protocol the message is either propagated to the "next" (i.e. immediately closer to the sink) ring sector, T_{i-1}, or is directly propagated to the sink. Consequently, after a maximum of i steps, the message will finally reach the sink. □

We note that we guarantee that the energy per sensor is the same on the average; we plan to study in future work energy balance properties guarantees with high probability as well, by taking into account the distribution (both spatial and in time) of event generation.

16.6 A Closed Form for the Forwarding Probability

Under specific assumptions (that we discuss and motivate below) we can make the calculation of probabilities p_i simpler. Combining Lemma 5 and Lemma 7 we have that

$$E[h_i] = \lambda_i N + E[f_i] = \frac{2i-1}{n^2}N + E[f_i] \tag{16.6}$$

By the corresponding relation for $E[h_{i-1}]$ it must be

$$\frac{\frac{2i-1}{n^2}N + E[f_i]}{(2i-1)\frac{\phi}{2}R^2}\left[p_i + (1-p_i)i^2\right]R^2 =$$

$$= \frac{\frac{2i-3}{n^2}N + E[f_{i-1}]}{(2i-3)\frac{\phi}{2}R^2}[p_{i-1} + (1-p_{i-1})(i-1)^2]R^2$$

But $2i-1 \simeq 2i-3$ and $\frac{\phi}{2}$, R^2 cancel. Dividing by N we get

$$\left[1 + \frac{E[f_i]}{N}\right]\left[p_i + (1-p_i)i^2\right] \simeq$$
$$\simeq \left[1 + \frac{E[f_{i-1}]}{N}\right]\left[p_{i-1} + (1-p_{i-1})(i-1)^2\right]$$

If $E[f_i] \simeq E[f_{i-1}]$ then the previous relation becomes

$$p_i + (1 - p_i)i^2 = p_{i-1} + (1 - p_{i-1})(i - 1)^2 \tag{16.7}$$

By solving the recurrence (see [9]), we get the following theorem.

Theorem 5 *If $E[f_i] \simeq E[f_{i-1}]$, $3 \leq i \leq n$, then the one-hop forwarding probability guaranteeing energy balance is*

$$p_i = 1 - \frac{3x}{(i + 1)(i - 1)}$$

where $p_2 = x \in (0, 1)$ a free parameter and $p_1 = 0$.

We remark that the assumption $E[f_i] \simeq E[f_{i-1}]$ is quite reasonable and well motivated. We provide the following intuitive explanation of why this happens. We remark indeed that the area sizes of adjacent sectors (and thus the number of events generated in such sections) are more or less the same, especially when i increases. Furthermore, the probability p_i of forwarding to the adjacent (towards the sink) sector increases very fast with i and becomes 1 in most sectors of the network (in the middle territory). Thus, the expected number of messages forwarded to adjacent sectors seems to be almost the same.

In Fig. 16.2, we graphically display both functions of p_i (i.e. the general solution and that obtained by the simple closed form). We can remark that the curves are very similar in shape and very close to each other. In particular in the "middle" territory (i.e. for $50 \leq i \leq 950$ when $n = 1000$) the two curves almost coincide.

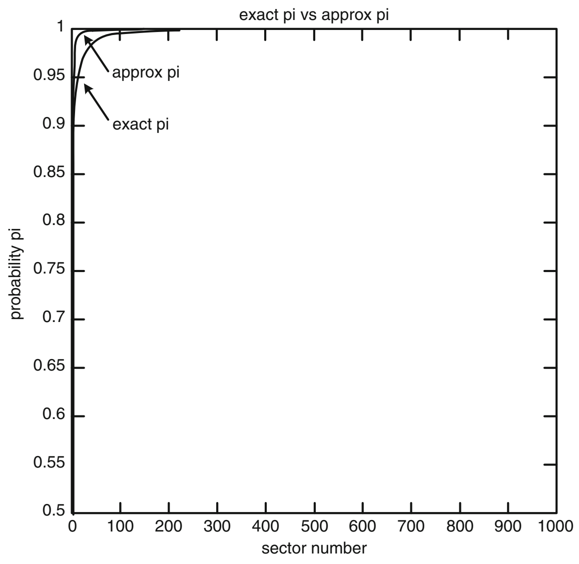


Fig. 16.2 Representation of exact p_i and approximated p_i

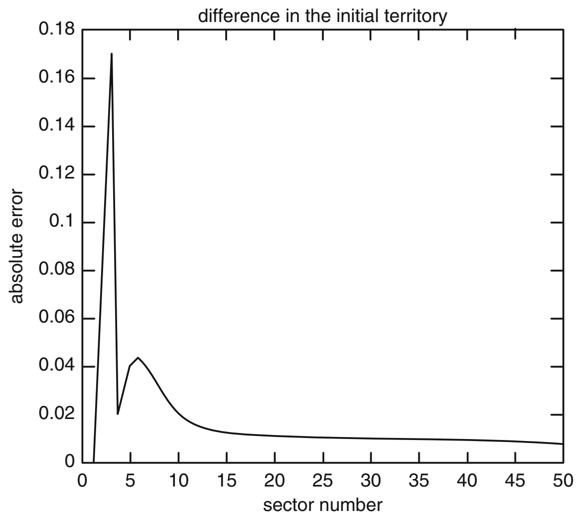


Fig. 16.3 "Absolute Error" in the first territory

For $1 \leq i \leq 50$ (in the initial territory) the exact and approximation probability is in most sections less than 0.08 (Fig. 16.3) apart and only in very few sectors the difference becomes 0.16. Finally, at the third territory ($950 \leq i \leq 1000$) the accurate solution and the approximation differ by less than 0.08 (Fig. 16.4). This shows that our solution under the simplifying assumptions is very tight (Fig. 16.5).

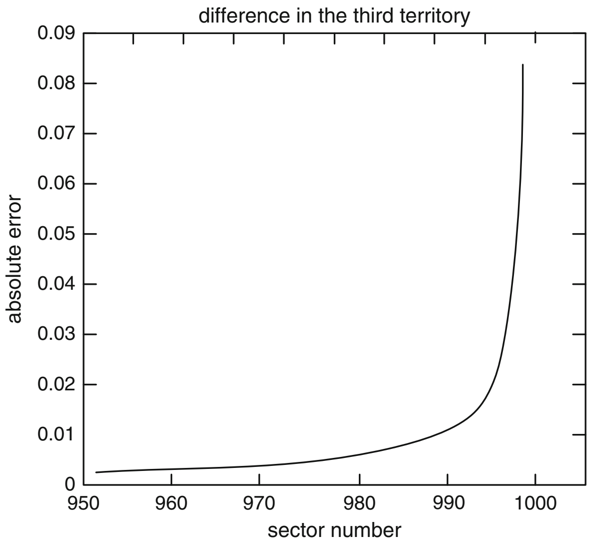


Fig. 16.4 "Absolute Error" in the third territory

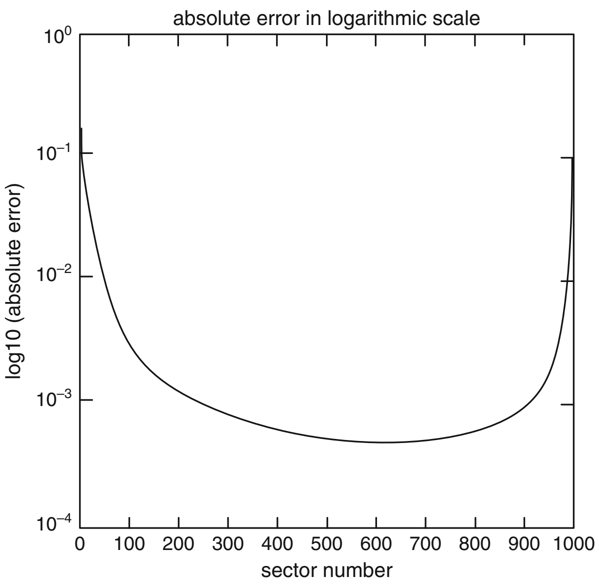


Fig. 16.5 "Absolute Error" in logarithmic scale for the whole network

16.7 A Generalized Algorithm

To enlarge the scope of the energy-balanced propagation protocol, we consider a more general model by removing the geometrical assumption that the sensors are uniformly distributed. Instead, we assume that the total energy budget of sensors located in the ring sector T_i is b_i. The price to pay to deal with this more general case is that we are no longer able to provide a closed form solution, like in Sect. 16.6, which leads easily to a distributed implementation. The worst case complexity of the algorithm is $\mathcal{O}(n^3)$ where n is the number of ring sectors in the network. As the presentation of the algorithm shows, this is due to the fact that sensors need to know all the network's parameters (message generation rate and energy budget of all the ring sectors).

However, the algorithm is able to compute some related solution to the problem even in the case when no energy-balanced solution exists. In that case, the solution produced by the algorithm is optimal in the sense that the lifetime of the network is maximized if we define the lifetime as follows:

Definition 12 (network lifetime) The lifetime of the sensor network is the minimal lifetime of the ring sectors $T_i,\ 1 \le i \le n$.

The lifetime of the ring sector T_i is estimated by $b_i / \left(p_i h_i c R^2 + (1 - p_i) h_i (i R)^2\right)$. Indeed, h_i is the number of messages handled by the ring sector T_i while we assume that N messages are generated. Then, h_i is proportional to the number of messages handled in each unit of time and the fraction is proportional to the total energy budget divided by the energy consumed in each unit of time. In the following, we introduce the notation

$$P_i = p_i h_i c R^2 + (1 - p_i) h_i (iR)^2$$

We start by considering a particular situation where messages are generated only in the ring sector T_k, and we look for a propagation strategy that leads to an even energy consumption in the ring sectors $T_i,\ 1 \leq i \leq k$. We point out that it is not possible to balance the energy consumption in the ring sectors $T_i,\ k < i \leq n$. Let us denote $\varepsilon_i,\ 1 \leq i \leq k$ the probability to forward one-hop further from ring sector i to $i - 1$. The mean energy consumption per message handled in ring sector i is

$$\varepsilon_i c R^2 + (1 - \varepsilon_i) c (iR)^2$$

and the number of messages allowed with the energy budget b_i is

$$\frac{b_i}{cR^2 \varepsilon_i + c(iR)^2 (1 - \varepsilon_i)} \tag{16.8}$$

Among this total number of messages, an ε_i fraction is forwarded to the next ring sector. So, to ensure energy-balanced consumption, the parameters ε_i have to satisfy

$$\varepsilon_i \frac{b_i}{cR^2 \varepsilon_i + c(iR)^2 (1 - \varepsilon_i)} = \frac{b_{i-1}}{cR^2 \varepsilon_{i-1} + c((i-1)R)^2 (1 - \varepsilon_{i-1})} \tag{16.9}$$

By taking into account that $\varepsilon_1 = 1$, we use Eq. (16.9) to compute the parameters $\varepsilon_i,\ 1 < i \leq k$ recursively. We notice that the computed parameters satisfy $\varepsilon_i \geq 0,\ 1 < i \leq k$, but it may happen that $\varepsilon_i > 1$ a situation that we consider later.

The idea of the algorithm is the following. We start by inspecting the messages generated in the first ring sector; we expect $N\lambda_1$ messages and this leads to an expected energy consumption $N\lambda_1 cR$. We compute the *power to battery ratio* by normalizing this mean energy consumption with the energy budget b_1, i.e. we consider $(N\lambda_1 cR)/b_1$, see the phase 1 in Fig. 16.6.

Next, we consider the messages generated in the second ring sector; the total expected number is $N\lambda_2$. Among these messages, we select J_2 that are sent directly to the sink and such that

$$\frac{b_2}{J_2 c (2R)^2} = \frac{b_1}{N\lambda_1 cR} \tag{16.10}$$

The next $\lambda_2 N - J_2$ messages are forwarded to the first ring sector with probability ε_2, with ε_2 defined by formula (16.9) (with $i = 2$). This ensures that the ratio of the power dissipated to the energy budget is the same in the first two ring sectors. Notice that if we interpret λ_i as the rate of messages generated per unit of time, this amounts to say that both ring sectors are running out of energy at the same time. Finally, notice that the probability that a message is forwarded to the first ring sector from the second one is

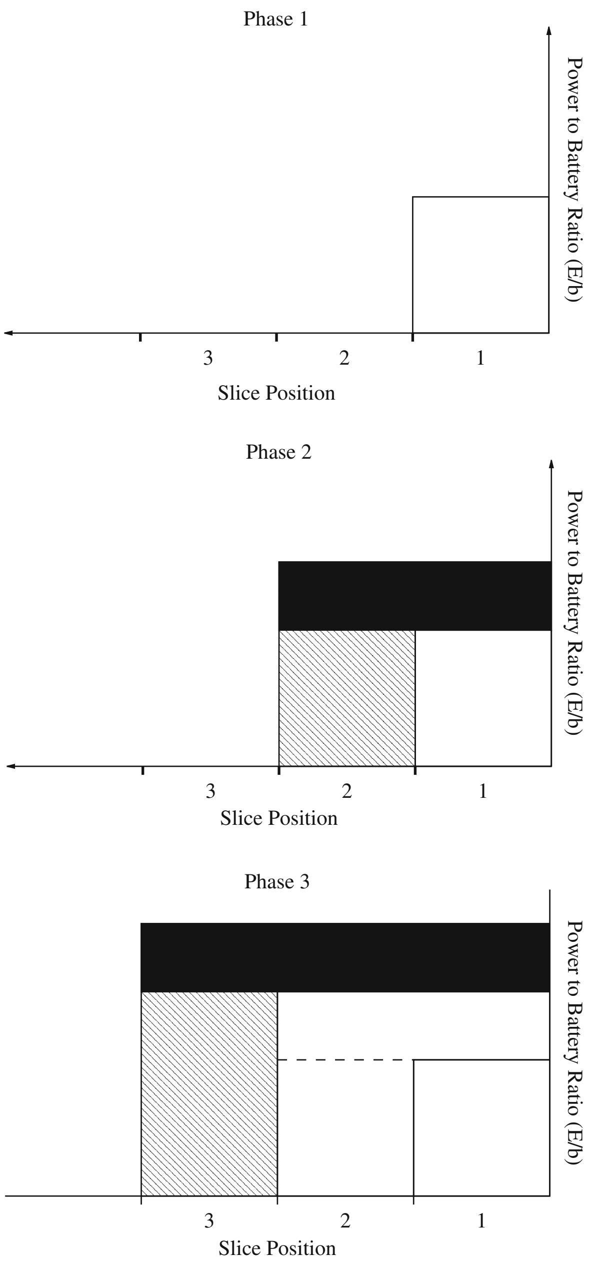

Phase 1
Power to Battery Ratio (E/b)
3
2
1
Slice Position
Phase 2
Power to Battery Ratio (E/b)
3
2
1
Slice Position
Phase 3
Power to Battery Ratio (E/b)
3
2
1
Slice Position

Fig. 16.6 Intuitive idea of the algorithm

$$\frac{\varepsilon_2 J_2}{N}$$

To continue, we select successively all the ring sectors and proceed accordingly. The process is shown in Fig. 16.6.

If the algorithm can run in this way, the complexity of the algorithm is $\mathcal{O}(n^2)$, see [19]. However, there are two difficulties that prevent the algorithm from running straightforwardly. The value ε_i computed with the help of formula (16.9) can be larger than 1 or the value J_2 (in general J_i) computed with Eq. (16.10) can be larger than $\lambda_2 N$ ($\lambda_i N$ in general).

The case when $\varepsilon_i > 1$ indicates that the energy budget of the ring sector i is not large enough to balance the energy consumption with the next ring sectors. In the model we describe in Sect. 16.2, this may happen if the number of sensors in the slice i is not large enough. Basically, the algorithm presented in [19] deals with this situation by enforcing $\varepsilon_i = 1$. However, as messages are forwarded to the next slices, the algorithm takes care of not leading to a situation where $\varepsilon_i > 1$ by limiting the total number of messages forwarded.

The second case corresponds to the situation where a ring sector T_i does not generate enough messages to equalize the energy consumption with the next ring sectors. The algorithm proceeds by looking to ring sectors $T_k,\ k > i$ and forwards the necessary messages to the ring sector T_i.

In both situations, the complexity of the algorithm is $\mathcal{O}(n^3)$, which corresponds to the worst case.

Unfortunately, it may happen that the algorithm does not provide an energy balance solution. This happens when all the strategies fail. We denote by P_i the mean energy consumption in the ring sector i (see Eq. (16.8))

$$P_i = p_i c R^2 + (1 - p_i) c (ir)^2$$

where p_i is the value computed by the algorithm and, $\widetilde{P}_i,\ 1 \le i \le n$, the mean energy consumption computed for some others values $\tilde{p}_i,\ i \le i \le n$. With these notations, it can be proved ([19]) that there always exists an index l such that

$$\frac{P_l}{b_l} \ge \frac{\widetilde{P}_l}{b_l}$$

This means that there does not exist a solution which improves the lifetime of all the ring sectors or equivalently, the solution computed by the algorithm is a (weak) Pareto optimal solution.

Moreover, it is proved in [19] that the solution provided by the algorithm satisfies

$$\min\{b_i/P_i\} \ge \min\{\tilde{b}_i/\widetilde{P}_i\}$$

Since Definition 12 of the network's lifetime is equivalent to computing $\min\{b_i/P_i\}$, the result shows that the solution computed by the algorithm maximizes the lifetime of the network.

An alternative solution of dealing with the problem described in this section is to consider the problem as a multiobjective optimization problem, i.e. to look for a solution maximizing the set of equations $\{P_i/b_i\}_i$. Multiobjective problems have rarely a unique solution, and a classical way of dealing with these problems is to search for Pareto optimal solutions [17].

16.7.1 A Remark About the Underlying Assumption

The model introduced in Sect. 21.2 assumes that the sensors are located uniformly in the ring sectors and it is expected that the number of messages handled by sensors belonging to the same ring sector will be equivalent. However, numerical simulations show that sensors that are located far away from the border of the next (towards the sink) ring sector consume more energy than the ones situated closely. This is due to the fact that sensors close to the border of the next ring sector have less incoming links than the ones located farther, see the left part of Fig. 16.7.

To deal with this uneven energy consumption, a spreading strategy is suggested in [19] to balance the energy consumption among sensors belonging to the same ring sector. Instead of transmitting a message directly to the sink, sensors take advantage of the possibility to transmit the message to a sensor belonging to the same ring sector that will, in turn, transmit to the sink.

Numerical validation of the spreading technique is displayed in Fig. 16.8 where we plot the radius of each of the 6280 sensors scattered in a 90 degrees sector divided into 20 ring sectors. The simulation runs for a total of 1, 18, 000 rounds.

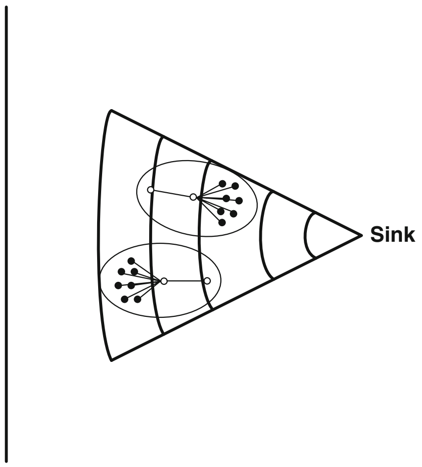


Fig. 16.7 Uneven distribution of the number of incoming links

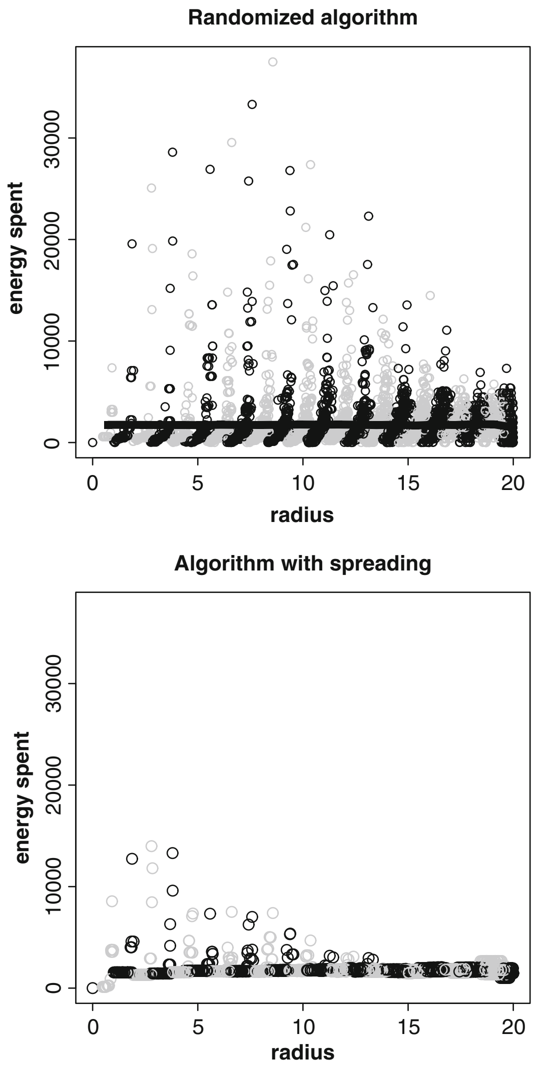


Fig. 16.8 Uneven distribution of the number of incoming links

In the upper part of Fig. 16.8 we observe that energy consumption increases with the radius, then decreases suddenly to a minimal value and continues to follow this pattern repeatedly. Actually, we observe that as we are close to the next ring sector the energy consumption is close to a minimal value and that the energy increases as the distance from the border increases.

In the lower part of Fig. 16.8 we observe the result produced with the spreading technique. We observe that the spreading technique equalizes the energy consumption among sensors belonging to the same ring sector. Besides these remarks, it is fair to conclude that the numerical simulations validate the good performance of the protocol for most of the sensors composing the network.

16.8 On the Optimality of Energy-Balance Protocols

In the preceding sections, we consider energy-balance data propagation algorithms. This particular class of protocols was first introduced with the motivation that they make the lifetime of the network longer. The next result that we present in this section is more surprising. Indeed, we show that besides prolonging the lifetime of the network, energy-balance propagation protocols maximize the flow of data in the network [10, 14]. In turn, this result supports Definition 12 of the network's lifetime. Actually, it is intuitive that the longer the network's lifetime the more the messages are handled.

To be more precise, the result that we discuss in this section is that if there exists probabilities $p_i, i \leq i \leq n$, such that the protocol balances the energy among the ring sectors then the flow is maximal and adding any link in the communication network does not increase the flow. It is relevant to emphasize that besides the fact that the flow is maximized by energy-balance protocols, the communication strategy that is described in Sect. 21.2 (sensors send to the next ring sector towards the sink or directly to the sink) is optimal. Rephrased in another way, provided that the protocol is energy balanced, only two levels of transmission power are sufficient to maximize the flow of data in the network.

The model of the network that we use in this section is the one introduced in Sect. 21.2 that we generalize by introducing the energy budget b_i, $1 \leq i \leq n$, available in ring sectors T_i, $1 \leq i \leq n$, similar to Sect. 16.7. Moreover, we assume that the distances are normalized in such a way that the energy cost of transmitting one ring sector away costs 1 unit of energy. This means that sending a message j ring sectors away costs j^2 units of energy.

In our setting, the total flow of messages in the network is the total flow getting to the sink. If we denote the flow by T, looking for the maximal flow in the network amounts to solve the linear program (16.11):

$$\begin{array}{rll} \max\ T, \text{ such that} & & \\ f_{0,i} = T\lambda_i & & 1 \leq i \leq n \\ \sum_{i=0}^{n} f_{i,j} = \sum_{i=0}^{n} f_{j,i} & & 1 \leq i \leq n \\ \sum_{i=1}^{n} f_{i,j}(i-j)^2 \leq b_i & & 1 \leq i \leq n \end{array} \tag{16.11}$$

In [14], this problem is referred as the *generalized flow maximization problem*. This problem considers all the possible flows (line 3) with the constraints that the energy budgets are respected (line 4) and that messages are generated in ring sectors T_i with probability λ_i. By restricting the flow to the topology described in Sect. 21.2, the optimization problem reads

$$
\begin{array}{ll}
\max\ T,\ \text{such that} & \\
f_{0,i} = T\lambda_i & 1 \le i \le n \\
\sum_{i=0}^{n} f_{i,j} = \sum_{i=0}^{n} f_{j,i} & 1 \le i \le n \\
\sum_{i=1}^{n} f_{i,j}(i-j)^2 \le b_i & 1 \le i \le n \\
f_{i,j} = 0 & 1 \le i \le N,\ j \notin \{0, j-1\}
\end{array}
\tag{16.12}
$$

The problem (16.12) is called *the mixed-flow maximization problem* in [14] since the routing algorithm *mixes* single-hops and multi-hops. We notice that problems (16.11) and (16.12) describe the routing algorithm by considering that messages emerge virtually from the sink node 0 to be sent to the ring sector T_i with probability λ_i (line 2) to be conveyed back towards the sink.

With this setting, we formally define the notion of *energy-balance* flow.

Definition 13 (energy balance) A flow $F = (T, f_{i,j})$ is energy-balanced if

$$\sum_{j=0}^{n} f_{i,j}(i-j)^2 = b_i,\ 1 \le i \le n$$

The definition stipulates that energy-balanced flows exhaust the energy budget of all the ring sectors simultaneously.

Let us consider the particular path where a message is generated in the ring sector T_i, forwarded towards the ring sector T_{i-1} and finally transmitted towards the sink. The total cost for conveying this message from the ring sector T_i to the sink is $1+(i-1)^2$, while transmitting directly towards the sink from the ring sector costs i^2 units of energy. Simple computation shows that the scaled total energy consumption of both paths is the same, i.e.

$$\forall i,\quad \frac{1}{i(i+1)} + \frac{(i-1)^2}{i(i-1)} = \frac{i^2}{i(i+1)}$$

This result can be generalized to every mix-flow path. If a message is generated in ring sector i and transits through ring sectors $i-1, i-2, \dots, i-k$ to the sink, then the total scaled energy cost satisfies

$$\frac{1}{i(i+1)} + \frac{1}{(i-1)i} + \frac{1}{(i-2)(i-1)} + \cdots + \frac{(i-k)^2}{(i-k)(i-k+1)} = \frac{i^2}{i(i+1)} \tag{16.13}$$

With the help of Eq. (16.13) we prove that energy-balanced flows are maximal. We denote $F = (T, f_{i,j})$ as an energy-balanced mixed-flow and we compute the total scaled cost of conveying messages generated from the ring sector i towards the sink. Equation (16.13) says that this total scaled cost is

$$T\lambda_i \frac{i^2}{i(i+1)}$$

The factor $T\lambda_i$ is the number of messages generated in ring sector i. By proceeding similarly for each ring sector we get the total scaled energy cost for the flow $F = (T, f_{i,j})$ as

$$\sum_{i=1}^{n}\sum_{j=0}^{n} f_{i,j}\frac{(i-j)^2}{i(i+1)} = T\sum_{i=1}^{n}\lambda_i\frac{i^2}{i(i+1)} \tag{16.14}$$

Notice that $f_{i,j} = 0$ if $j \notin \{0, i-1\}$ because the flow is solution to (16.12).

We consider a second mixed flow $\bar{F} = (\bar{T}, \bar{f}_{i,j})$. Equation (16.14) is still valid for the flow $\bar{T}$ but, because the flow is not necessarily energy balanced, we have

$$\sum_{j=0}^{n}\bar{f}_{i,j}(i-j)^2 \leq b_i,\ 1 \leq i \leq n$$

Using this last equation and (16.14), we obtain

$$T\sum_{i=1}^{n}\lambda_i\frac{i^2}{i(i+1)} = \sum_{i=0}^{n}\frac{b_i}{i(i+1)} \geq \sum_{i=1}^{n}\sum_{j=0}^{n}\bar{f}_{i,j}\frac{(i-j)^2}{i(i+1)} = \bar{T}\sum_{i=1}^{n}\lambda_i\frac{i^2}{i(i+1)}$$

The extreme terms of the inequality show that $T \geq \bar{T}$. So, we finally have proved the following:

Theorem 6 *(Jarry et al. [14]) Energy-balanced mixed-flows flows are solutions to the mixed-flow maximization problem, i.e. they are maximal.*

Remark 2 We emphasize the meaning of the result. Energy-balance flows are motivated by the need of maximizing the lifetime of the sensor network. The result shows that unexpectedly, such flows are maximal. Moreover, the result offers a convenient way of maximizing the flow in a distributed environment. Indeed, the energy-balance property is a *local* characteristic of the flow since it is enough to show that each sensor balances the energy consumption with its neighbouring sensors to ensure that the flow is globally balanced (provided that the network is connected). The locality of the property allows for distributed implementation, an aspect that we discuss later. We also point out here that the characterization of the conditions (the λ_i) ensuring the existence of an energy-balance flow is still an open question.

Actually, we can prove that energy-balance mix-flows are maximal even if we enlarge the class of flows, i.e. energy-balanced solutions of (16.12) are maximal among the solutions of (16.11).

For a general flow, simple computation shows that

$$\forall i,\quad \frac{(i-j)^2}{i(i+1)} + \frac{j^2}{j(j+1)} \geq \frac{i^2}{i(i+1)}$$

More generally, for some path $i, i_2, \ldots, i_k$ (compare with (16.13)), we get that the scaled total cost of conveying a message from the ring sector i to the sink satisfies

$$\frac{(i - i_2)}{i(i+1)} + \frac{(i_2 - i_3)^2}{i_2(i_2+1)} + \cdots + \frac{i_k^2}{i_k(i_k+1)} \geq \frac{i^2}{i(i+1)} \tag{16.15}$$

Decomposing flows into paths enables us to deduce that for a general flow $\bar{F} = (\bar{T}, \bar{f}_{i,j})$ and an energy-balanced mix-flow $(F = (T, f_{i,j}))$ it is:

$$T \sum_{i=1}^{n} \lambda_i \frac{i^2}{i(i+1)} = \sum_{i=1}^{n} b_i \frac{1}{i(i+1)} \geq \sum_{i=1}^{n} \sum_{j=0}^{n} \bar{f}_{i,j} \frac{(i-j)^2}{i(i+1)} \geq \bar{T} \sum_{i=1}^{n} \lambda_i \frac{i^2}{i(i+1)}$$

This last inequality shows that $T \geq \bar{T}$ and the energy-balance mixed-flow is then maximal. Notice that the inequality (16.15) is strict if there are successive indices such that $| i_k - i_{k+1} |> 1$. This shows that an energy-balance mixed-flow cannot be increased by adding edges to the communication graph. Thus, we get the following theorem:

Theorem 7 *(Jarry et al. [14]) An energy-balance solution to the mix-flow maximization problem (16.12) is also a solution to the general flow maximization problem (16.11).*

16.8.1 Learning the Protocol's Parameters

Given the protocol's parameters λ_i, $1 \leq i \leq n$, we have discussed the computation of the appropriate values for the probability of transmitting to the next ring sector p_i, $1 \leq i \leq n$. To take into account the situations where these parameters are not known, we present here a learning strategy [16, 17].

We start by redefining the necessary key equations. Given that a ring sector generates a message, we compute the probability that the message is handled by the ring sector T_i. This happens if the message is generated in T_i or in the ring sector T_{i+1} and forwarded to T_i or so on up to the ring sector n. Then, the probability x_i that a generated message is handled by the ring sector T_i is given by

$$x_i = \lambda_i + \lambda_{i+1} p_{i+1,i} + \lambda_{i+2} p_{i+2,i+1} p_{i+1,i} + \cdots + \lambda_n p_{n,n-1} p_{n-1,n-2} \cdot \ldots \cdot p_{i+1,i}$$

A message induces a mean energy consumption in T_i which is given by

$$x_i \left(p_i \frac{1}{b_i} + (1 - p_i) \frac{i^2}{b_i} \right)$$

where we normalize the energy consumption by the energy budget b_i since we search for probabilities p_i that balance the energy consumption. Actually, the energy-balance condition can be restated as.

$$x_i \left(p_i \frac{1}{b_i} + (1 - p_i) \frac{i^2}{b_i} \right) = \lambda, \ \forall 1 \le i \le n$$

Solving this equation for p_i leads to

$$p_i = \frac{i^2}{i^2 - 1} - \frac{b_i}{(i^2 - 1)x_i} \lambda \tag{16.16}$$

We observe that the probabilities x_i satisfy the recursion $x_i = p_{i+1}x_{i+1} + \lambda_i$. Then, it is possible to compute the probabilities p_i starting for ring sector T_n up to T_1 provided that the value of λ is known. The value of λ is given by

$$\lambda = \frac{x_1}{b_1}$$

By using the recursion $x_i = p_{i+1}x_{i+1} + \lambda_i$ and Eq. (16.16) we get that λ satisfies

$$\lambda = f(\lambda) = a + b\lambda \tag{16.17}$$

with

$$b_1 a = \lambda_1 + \lambda_2 C_2 + \lambda_3 C_3 C_2 + \cdots + \lambda_n C_n C_{n-1} \ldots\ldots C_2$$

$$b_1 b = -D_2 - D_3 C_2 - D_4 C_3 C_2 - \cdots - D_n C_{n-1} \ldots\ldots C_2$$

$$C_i = \tfrac{i^2}{i^2-1}, \quad D_i = \tfrac{b_i}{i^2-1}, \quad 2 \le i \le n$$

Because $a > 0$, $b < 0$, the linear equation (16.17) has a unique solution. This proves that provided there are probabilities p_i balancing the energy consumption, the solution is unique.

We consider the situation where the parameters λ_i are not known. To simplify the expression, we assume $b_i = 1$, the general case following straightforwardly. Looking at Eq. (16.16), we observe that the probabilities that a message is handled by the ring sectors x_i are sufficient for computing the probabilities p_i (remember that $\lambda = x_1$). By observing the paths followed by the different messages, the sink can estimate these probabilities. Moreover, since it is usual to assume that the sink is not energy limited, it is reasonable that the sink transmits the estimate to the sensors. The implementation of the learning strategy requires that sensors add the number of ring sectors they belong to as they handle a message.

The sink estimates the probabilities x_i by looking at the path followed by each message. A first attempt would be to compute the ratio of the number of messages that passed through the ring sector i over the total number of received messages. However, with such a solution the protocol uses a poor estimate of the p_i for a long time before the sensors can compute a better estimate by using the x_i's values transmitted by the sink. We then use a recursive estimation method. The estimates x_i are updated after the arrival of each message, see Fig. 16.9 for the pseudocode.

```
Initialise $\tilde{x}_0 = \lambda, \ldots, \tilde{x}_n$
Initialise NbrLoop=1
repeat forever
  Send $\tilde{x}_i$ values to the stations which compute their $p_i$ probability
   wait for a data
   process the data
  for i=0 to n
    if the data passed through slice $i$ then
      $X \leftarrow 1$
    else
      $X \leftarrow 0$
    end if
     Generate $R$ a $\tilde{x}_i$-bernoulli random variable
     $\tilde{x}_i \leftarrow \tilde{x}_i + \frac{1}{NbrLoop}(X - R)$
     Increment $NbrLoop$ by one.
  end for
end repeat
```

Fig. 16.9 Pseudocode for estimation of the x_i value by the sink

Recursive estimation is relevant in our context since the probability p_i can be updated online. This ensures that the protocol does not waste energy by using a poor estimate for a long time. The convergence time of the method is hard to compute and bounds are overestimated, typically of order $\mathcal{O}(1/\sqrt{NbrLoop})$. However, it is a well-known result that the estimated values quickly approximate the right value and the estimate shows damped oscillation around the true value.

We display in Fig. 16.10 some numerical validations of the learning strategy. We choose the parameters $\lambda_i = (2i-1)/n^2$, equal energy budget b_i and networks with 3, 10, 20, 30 ring sectors. We plot the figure of the successive approximation of the value $\lambda = x_1$. The probabilities are all initialized with the value 0.5.

We observe in Fig. 16.10 that the estimation quickly reaches a value close to the final one and progressively refines the value. Although the total time of convergence appears to be long, the estimates are quickly of the right order of magnitude.

16.8.2 A Simple Distributed Strategy

In this last section, we present a very simple strategy which consists in forwarding to the next ring sector only if the current energy consumption is lower and sending directly to the sink in the other case. The implementation of this strategy requires to know the energy consumption level of the next ring and this does not require any extra message. Indeed, while a sensor transmits a message, it includes a supplementary data that indicate the current level of energy consumption. Since the message is

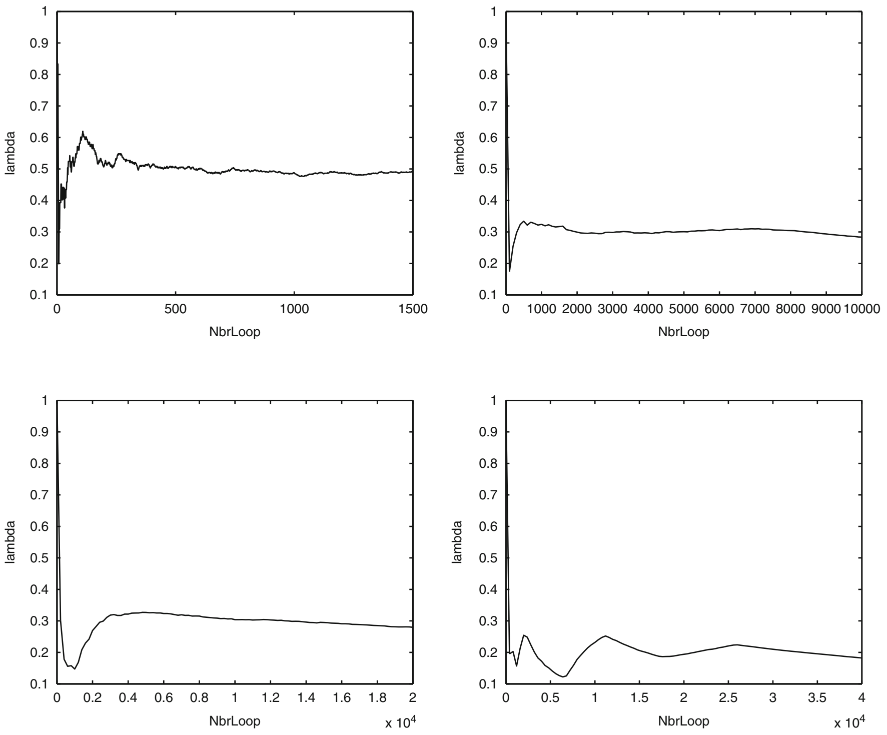


Fig. 16.10 Numerical validation of the learning strategy. On the *left* of the *top* figure a network with three ring sectors and, on the *right* 10 ring sectors. On the *left* of the *bottom* figure a network with 20 ring sectors, and, on the *right* 30 ring sectors

received by all the sensors belonging to adjacent ring sectors, they can be aware of the current level of energy consumption.

The analysis of the simple strategy is based on the Markovian character of the process. We denote by $x_i(t)$ the energy consumption of the ring sector i at time t. Since we are only interested in the transmission of messages, we assume that the time is discrete, $t = 0, 1, 2, \ldots$ and that each time instant corresponds to the transmission of a message. The discrete dynamics is expressed as a map

$$\begin{pmatrix} x_n(t) \\ x_{n-1}(t) \\ \vdots \\ x_1(t) \end{pmatrix} \Longrightarrow \begin{pmatrix} x_n(t+1) \\ x_{n-1}(t+1) \\ \vdots \\ x_1(t+1) \end{pmatrix}$$

Actually, since our main concern is to balance the energy consumption, it is more convenient to work with the reduced state vector given by

$$X(t) = \begin{pmatrix} x_n(t) - x_{n-1}(t) \\ x_{n-1}(t) - x_{n-2}(t) \\ \vdots \\ x_2(t) - x_1(t) \end{pmatrix}$$

The (reduced) state vector changes as a new message is sent from one ring sector. A new message is generated with probability λ_i. The discrete dynamics can be assumed Markovian, since the evolution of the state vector depends only on the current state.

We can prove that the Markov chain is *irreducible* [14] by showing that from any state the system reaches the null state $X(t) = (0, 0, \ldots, 0)$ with positive probability. This is important because, if the chain is not irreducible, the long-time dynamic might depend on the initial state.

The simple strategy aims at balancing the energy consumption. With our formulation this amounts to expect that $X(t) \approx (0, 0, \ldots, 0)$, $\forall t$ sufficiently large. It is clear that this is a too strong requirement. We then require that the Markov chain is *stable* in the following sense:

Definition 14 (stability) The Markov chain $\{X(t)\}_{t\geq 0}$ is stable if there exists a neighborhood D of the origin which is positive recurrent. This means that

$$P\big(X(t) \in D,\ t < \infty\big) = 1$$

The statement of the next theorem contains sufficient conditions ensuring the stability of the Markov chain.

Theorem 8 *[14] We assume that $\lambda_i > 0,\ 1 \leq i \leq n$. Then the set*

$$D = \left\{ X = (X_n, \ldots, X_2)\ s.t.\ |\ X_i\ | \leq \frac{i^2}{2},\ i = 2, \ldots, n \right\}$$

is positive recurrent if

$$i^2 \lambda_i > (i-1)^2 \lambda_{i-1},\ i = 2, \ldots, n$$

Before discussing informally how the result is proved, we consider more closely the conditions ensuring the stability. If we impose the conditions $\lambda_i i^2 = \lambda_{i-1}(i-1)^2$, it is possible to balance the energy consumption if all ring sectors transmit directly to the sink. However, with these conditions our proof of the stability is not valid and we cannot conclude. Anyway, let us think that the frontier between conditions ensuring the stability or the instability of the strategy might be well characterized by direct transmissions towards the sink.

To proceed further, let us assume that $\lambda_n = 0$. In this case, it is clear that the energy consumption of the last ring sector T_n cannot be balanced since no messages are generated in T_n and T_n has no way of receiving messages.

Let us then assume that $\lambda_i = 0, i < n$. The protocol ensures that $x_{i+1}(t) - x_i(t) < i^2$. The bound is attained in the case where $x_{i+1}(t) < x_i(t)$ and a message is sent directly to the sink from T_{i+1}. A similar argument shows that $x_i(t) - x_{i+1}(t) < i^2$, so we finally have $| x_i(t) - x_{i+1}(t) | < i^2$.

The preceding reasoning can be expanded to the case where we have $\lambda_{i+j} = \lambda_{i+j-1} = \cdots = \lambda_i = 0$ to show that, provided $\lambda_{i+j+1} \neq 0$, the entries of the reduced state vector are bounded. Then, it seems reasonable to expect that the only pathological situation is when external ring sectors do not generate enough messages to balance the energy consumption with the lower ring sectors.

To conclude this section, we discuss informally the proof of the result which can be found in [14]. We use the function

$$f(x) = \sum_{i=2}^{n} | x_i(t) - x_{i-1}(t) |$$

With the sufficient conditions, we can prove that

$$E\big(f(X(t+1)) - f(X(t)) \mid X(t)\big) \leq -\varepsilon, \ \varepsilon > 0 \tag{16.18}$$

for all $X(t) \notin D$. This means that on average the value of the function f decreases outside the domain D. Formally, we have that $f(X(t))$ is a positive supermartingale outside of D. Since positive supermartingales converge, the orbit of $X(t)$ has to enter the domain D at some time. If not, the supermartingale will converge to a point and this is in contradiction with (16.18). Once the orbit of $X(t)$ is in D, the supermartingale property no longer holds, and the value of $f(x(t))$ is going to increase while the orbit of $X(t)$ is escaping from D. The process then evolves in this way, escaping D, returning to D, and so on. This shows that D is positive recurrent and that the discrete dynamics is stable.

16.9 Conclusions

We have studied a particular aspect of energy optimization in wireless sensor networks, that of energy balance, i.e. guaranteeing that all sensors in the network have the same energy throughout the protocol evolution. The importance of the energy balance property is based on the fact that it prolongs the network lifetime by avoiding an early disconnection of overused network regions.

For the energy balance problem we presented a few characteristic algorithms, assuming different levels of network knowledge in each case. Our algorithms basically use randomization in choosing the right data propagation pattern in order to appropriately handle the energy-latency trade-off and achieve energy balance. We finally provide formal evidence that energy balance leads at the same time to energy optimality.

Acknowledgments This work has been partially supported by the IST Programme of the EU under contract number IST-2005-15964 (**AEOLUS**).

References

1. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. In *the Journal of Computer Networks*, 38: 393–422, 2002.
2. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. In *the IEEE Communications Magazine*, 102–114, August 2002.
3. A. Boukerche, X. Cheng, and J. Linus. Energy-aware data-centric routing in microsensor networks. *ACM Madeling Analysis and Simulation of Wireless and Mobile Systems (MSWIM2003)*, Paris, France, pages 42–49, 2003.
4. I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis. Efficient and robust protocols for local detection and propagation in smart dust networks. In: *The ACM/Baltzer Mobile Networks and Applications* (MONET) *Journal, MONET* 10(1): pages 133–149, 2005.
5. I. Chatzigiannakis, T. Dimitriou, S. Nikoletseas, and P. Spirakis. A probabilistic algorithm for efficient and robust data propagation in smart dust networks. In: *The Ad-Hoc Networks Journal*, *Elsevier*, 4(5): 621–635, 2006.
6. I. Chatzigiannakis, T. Dimitriou, M. Mavronicolas, S. Nikoletseas, and P. Spirakis. A comparative study of protocols for efficient data propagation in smart dust networks. In *the Parallel Processing Letters* (PPL) *Journal*, 13(4) 615–627, 2003.
7. I. Chatzigiannakis and S. Nikoletseas. A Sleep-awake protocol for information propagation in smart dust networks. In: *Proceedings of the 3rd International IEEE Workshop on Mobile and Ad-hoc Networks* (WMAN), held in conjunction with IPDPS 2003, IEEE Press, 2003.
8. J.C. Dagher, M.W. Marcellin, and M.A. Neifeld. A theory for maximizing the lifetime of sensor networks. *IEEE Transactions on Communications*, 55(2):323–332, 2007.
9. C. Efthymiou, S. Nikoletseas, and J. Rolim. Energy balanced data propagation in wireless Sensor Networks. In *Wireless Networks (WINET) Journal*, 12(6): 691–707, 2006.
10. A. Giridhar, and P.R. Kumar. Maximizing the functional lifetime of sensor networks. In: *The Proceedings of the 4th international symposium on Information processing in sensor networks*, Los Angeles, USA, 2005.
11. W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In: *Proceedings 33rd Hawaii International Conference on System Sciences*—HICSS'2000, 2000.
12. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A scalable and robust communication paradigm for sensor networks. In: *Proceedings 6th ACM/IEEE International Conference on Mobile Computing*—MOBICOM'2000, 2000.
13. C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. Extended version of [18].
14. A. Jarry, P. Leone, O. Powell, and J. Rolim. An optimal data propagation algorithm for maximizing the lifespan of sensor networks. In *the Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems* (DCOSS'06), Lecture Notes in Computer Science 4026, Springer, Berlin, pages 405–421, 2006.
15. J.M. Kahn, R.H. Katz, and K.S.J. Pister. Next century challenges: Mobile networking for smart dust. In: *Proceedings 5th ACM/IEEE International Conference on Mobile Computing*, pages 271–278, Seattle, USA, September 1999.
16. P. Leone, S. Nikoletseas, J. Rolim. An adaptive blind algorithm for energy balanced data propagation in wireless sensors networks. *IEEE International Conference on Distributed Computing in Sensor Systems* (DCOSS 2005), Marina Del Rey, California, Lecture Notes in Computer Science, Springer, 2005.
17. P. Leone, S. Nikoletseas, and J. Rolim. Stochastic models and adaptive algorithms for energy balance in sensor networks. *In the Theory of Computing Systems* (TOCS) *Journal*, 2009.

18. S. Nikoletseas, I. Chatzigiannakis, A. Antoniou, C. Efthymiou, A. Kinalis, and G. Mylonas. Energy efficient protocols for sensing multiple events in smart dust networks. In smart dust networks. In: *Proceedings 37th Annual ACM/IEEE Simulation Symposium* (ANSS'04), IEEE Computer Society Press, pages 15–24, 2004.
19. O. Powell, P. Leone, and J. Rolim. Energy optimal data propagation in wireless sensor networks. *In the Journal of Parallel and Distributed Computing* (JPDC), 67(3): 302–317, 2007.
20. S. M. Ross. *Stochastic Processes, 2nd Edition*. Wiley, 1995.
21. C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Topology management for sensor networks: Exploiting latency and density. In: *Proceedings MOBICOM 2002*.
22. M. Singh, and V. Prasanna. Energy-optimal and energy-balanced sorting in a single-hop wireless sensor network. In: *Proceedings First IEEE International Conference on Pervasive Computing and Comminications*—PERCOM, Dallas-Fort Worth, Texas, USA, page 50, 2003.
23. P. Triantafilloy, N. Ntarmos, S. Nikoletseas, and P. Spirakis. NanoPeer networks and P2P worlds. In: *Proceedings 3rd IEEE International Conference on Peer-to-Peer Computing*, 2003.

Chapter 17
Dense, Concentric, and Non-uniform Multi-hop Sensor Networks

Sajal K. Das, Alfredo Navarra, and Cristina M. Pinotti

Abstract In this chapter, we consider a large-scale sensor network, in a circular field, modeled as concentric coronas centered at a sink node. The tiny wireless sensors, severely limited by battery energy, alternate between sleep and awake periods, whereas the sink is equipped with high transmission power and long battery life. The traffic from sensors to the sink follows multi-hop paths in a many-to-one communication pattern. We consider two fundamental and strictly related problems, the *localization* and the *energy hole* problems. We first survey on recent algorithms most extensively studied in the literature and summarize their pros and cons with respect to our assumptions. Then we present our solutions tailored for dense and randomly deployed networks. In our localization protocol, the sensors learn their coarse-grain position with respect to the sink, and hence the sink acts as a reference point for the network algorithms, in particular the routing algorithm. For this role of the sink, the network may incur in a special energy hole problem, known as the sink hole problem. From this perspective, the localization and energy hole problems are strictly related. Our solution for the energy hole problem adopts a non-uniform sensor distribution, compatible with the proposed localization solutions, that adds more sensors to the coronas with heavier traffic. In conclusion, we show that the network model under consideration can solve the localization and energy hole problems by properly tuning some network parameters, such as network density.

17.1 Introduction

A *wireless sensor network* (WSN) consists of a large number of distributed sensors which cooperate for collecting and disseminating information (see [2, 5, 58] for a survey on WSNs). Sensors usually monitor some local events like temperature, humidity, motion, and radiation. Combined with a microprocessor and a low-power radio transceiver, each of them represents a smart network-enabled node. Sensors are severely constrained in battery power and their longevity in the network

S.K. Das (✉)
Department of Computer Science and Engineering, Center for Research is Wireless Mobility and Networking (CReWMaN), University of Texas at Arlington, Arlington, TX, USA
e-mail: das@uta.edu

is crucial for the life span of the whole network. Energy-efficient architectures, algorithms, and protocols are at the basis of research in the field of WSNs. One of the most important strategies for prolonging the lifetime of the sensors is to allow them to switch off their transceiver as much as possible, thus alternating between sleep and awake periods according to the running protocol or to the occurrence of events.

Data collected by the sensors are sent via multi-hop paths to some sink nodes where data are processed and stored. The sink node might be some fixed infrastructure close to or around which the sensor network has been deployed; or it could be some temporary data collector which moves through the network for that purpose. In either case, in order to allow the sink to regenerate the map of the sensed events, sensors must relate the collected information to their position. The procedure by which a sensor gets known of its actual position is referred to as the *localization* process. The simplest but expensive way to provide sensors with such information is to empower them with global positioning system (GPS) receiver. However, this contradicts the requirement of inexpensive sensors with limited energy. Therefore many strategies have been proposed in the literature to overcome the use of special hardware [31, 45, 87]. Based on the requirements or the applications, the localization process might provide coarse- or fine-grain coordinates to the sensors at the expenses of different levels of energy consumption.

In this chapter, we are mainly interested in coarse-grain localization schemes that associate each sensor with two coordinates. We refer to a circular sensor field, centered at a sink node, subdivided into sectors and concentric coronas (see Fig. 17.1d). In the simplest setting, sectors cut the circular area into equiangular portions and the coronas have the same width. At the end of a localization process, each sensor must be aware of the sector and the corona where it resides. Clearly, special cases of this model can be easily defined and managed once the basic problem is solved. For example, coronas and sectors might span different areas; more than one sink node can be considered and/or mobile sink nodes; sensors might be of different types according to their capabilities, available energy, running protocols.

Once the localization procedure terminates, sensors are ready to accomplish their specific tasks and start to sense and route the collected data. Events and protocols determine some traffic in the network toward the sink nodes. Recently, it has been experimentally noticed and theoretically supported that sensors closer to the sink consume their battery rather faster than the others. This process tends to isolate the sink node from the rest of the network which becomes useless. Such a problem is referred to in the literature as the *energy hole* problem. A careful strategy to avoid such a side-effect would prolong the life span of the network and optimize its performances. The main idea behind the resolution of the energy hole problem is to propose a balanced consumption of the batteries among all the sensors. This helps to avoid the situation that a few sensors drain their batteries compromising the network behavior. In [16] for instance, the authors present a distributed algorithm that in each step probabilistically chooses between either forwarding data one-hop further toward the sink or sending data directly to the sink. This randomized design is meant to balance the one-hop transmissions (which are cheap regarding energy but

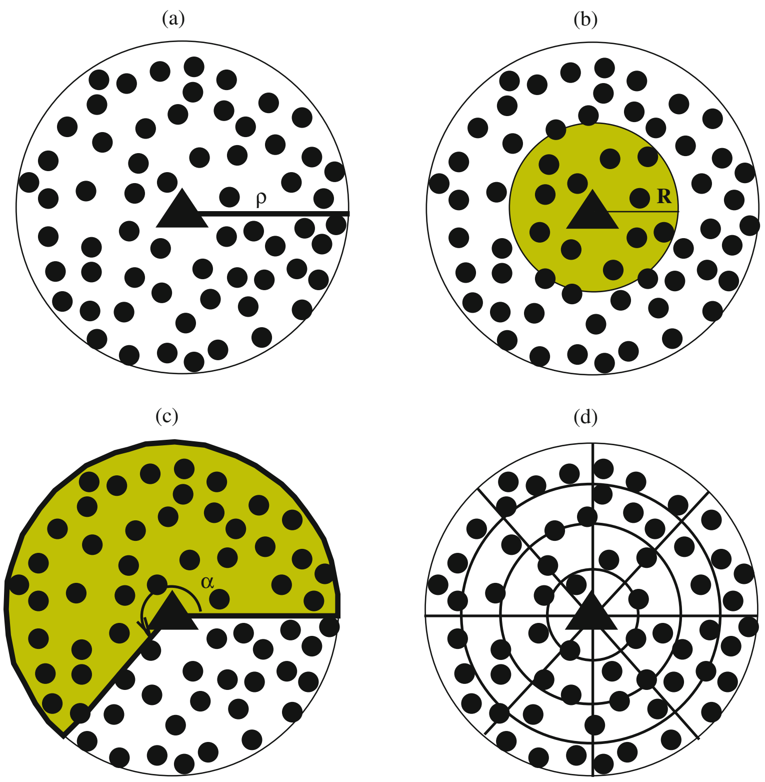


Fig. 17.1 A sink-centric subnetwork: (**a**) the sensors deployed in a circular field of radius ρ, (**b**) a sink broadcast of range R, (**c**) a sink broadcast of angle α, (**d**) the virtual coordinate system

need a lot of time) with the long, direct transmissions (which are more expensive and fast but bypass the critical region around the sink). This result is nicely generalized in [23, 42] where the authors provide adaptive methods for weaker models, formally show the optimality of this two-way propagation, and also investigate the deeper relation between energy balance and energy optimality.

A uniform consumption of the available energy in the network might also be accomplished by various approaches such as different node distributions and different communication protocols or heterogeneous sensors (see for instance [13, 38, 40, 53, 55, 60, 62] for different strategies). In this chapter, we present recent theoretical results in this field and also provide the most promising techniques to avoid energy holes.

The remainder of the chapter is organized as follows. Section 17.2 surveys the existing literature related to the localization and energy hole problems. Section 17.3 provides the model under consideration, the definitions and the underlying assumptions of our discussion. Section 17.4 describes our recent work on localization algorithms, while Sect. 17.5 summarizes our recent work on the energy hole problem. Finally, Sect. 17.6 provides concluding remarks and some hints for future works.

17.2 Related Work

Wireless sensor networks have been extensively studied in recent years. By reducing the deployment overhead in terms of unit cost or installation time, the development of dense distributed sensor systems becomes feasible and sensors can be placed closer to the phenomena they sense, thus yielding increased signal quality. Sensors are strongly coupled to the physical world, and their spatial positions, or *localization*, is a crucial factor in any task they perform [87]. As mentioned, an important factor that influences the system performance is the energy consumption. For example, as sensors in dense networks are generally powered by small inexpensive batteries, the life span of the whole network depends on the longevity of the sensors in the system. This suggests us to maintain the sensors activity and hence energy consumption as low as possible. The lifetime of sensors is usually divided into two phases that alternate: the awake and sleep periods [8, 13, 29]. In the former, sensors act with their full capabilities, while in the latter, only few operations are performed. Energy consumption also drives the communication protocols. Direct sensor-to-sink transmissions are the easiest way for reporting the sensed data to the external world, but the sensors farther away from the sink would run out of energy quickly due to the long transmission distance. Thus, to save energy, multi-hop routing schemes are more preferable, which, however, tend to overuse the intermediate sensors, particularly those close to the sink, thus leading to what are called energy holes around the sink.

In this chapter, we focus our attention on these two basic problems for sensor networks, the *localization* and the *energy hole* problems.

17.2.1 About Localization

In the literature, the task of determining the geographic position of sensors in the physical world, referred to as *localization*, is recognized as a fundamental problem in designing sensor networks and has been extensively studied (see, e.g., [5, 22, 31, 87]).

We restrict our attention to those systems in which the sensors are helpful during the localization process. Note that this is not always the case. For example, a system to localize animals which relies on sensors tied to the animal's neck cannot assume that the animal is acting with the intent to be localized, and thus sensors must be always awake with a considerable energy cost.

As indicated earlier, localization hardware, such as GPS for each sensor, is expensive in terms of both unit sensor cost and energy consumption especially when used in dense sensor networks. Prominent solutions assume the existence of anchor nodes which are aware of their location (because they are the only sensors provided with GPS) and allow other sensors to infer their locations by exchanging information with them. Such localization algorithms can be divided into three categories: *range-based* [15, 34], *range-free* [18, 45], and *coarse-grained* [2, 6– 9, 33, 39, 52, 56] localization schemes, as summarized below.

In range-based algorithms, sensors estimate their distance to anchors, using some specialized hardware. The technique to measure the distance anchor–sensor with minimum hardware requirement is the *received signal strength indication* (RSSI) because all sensors have a radio and can measure the strength of the received signal. Since the energy of a radio signal diminishes with the square of the distance from the signal's source, measuring the strength received by the sensor allows to calculate the source–sensor distance. However, this technique assumes regular radio propagation, while it is highly non-uniform and, thus, vulnerable to fading and occlusions [5]. The *time difference of arrival* (TDoA) measures the time required by a signal (say, sound) to traverse the distance from a localized source to a sensor. This method, which requires that sensors are provided with microphone, is extremely accurate (up to a few centimeters) when the sensor and the source have the line of sight [20]. Similar to TDoA, in *angle of arrival* (AoA), the phase difference of arrival is measured to get the anchor–sensor distance [4, 10, 14]. Such measurements are then applied to range-based methods, like triangulation, trilateration, which compute the position of a sensor by measuring distances from at least three reference anchors [4, 10, 15, 34, 36]. Another range-based algorithm is the centralized *multi-dimensional scaling* (MDS) which finds the locations of n sensors by computing their n^2 relative distances [5]. In the distributed *centroid* algorithm [11], the sensors simply localize themselves by averaging the positions of all anchors that their radio can receive. Despite simplicity of the method, the resulting localization is not accurate, especially when the number of anchors is low.

On the other hand, range-free algorithms do not use any special hardware to measure distance and accept a less accurate localization. Some methods, similar to distance vector routing, allow the sensors to find the number of hops from the anchors. Anchors flood their location throughout the network maintaining a running hop count at each sensor along the way. Sensors calculate their positions based on the received anchor locations, namely the hop count from the corresponding anchor, and the average distance per hop [35]. In [18], an iterative method is proposed to narrow down the position accuracy until a tolerable error in the positioning is reached. In practice, each sensor repeatedly chooses a triple of anchors from all audible anchors and tests whether it is inside the triangle formed by them, until all triples are exhausted or the required accuracy is achieved. At this point, the center of gravity of all of the triangles in which a sensor resides is assumed to be the sensor's estimated position. Interesting experimental results are shown in [14], where the positioning is obtained either by means of some anchors in the system or as a relative positioning without anchors. In practice, sensors become aware of the relative positions with respect to their distance to other sensors, but without obtaining real coordinates.

The localization algorithms discussed so far assume that the radio can receive multiple anchor nodes and that anchors are special sensors mainly because they know their spatial coordinates. Instead, several recent papers [2, 6– 9, 33, 39, 52, 56] have considered the localization problem in heterogeneous networks where a single node, called sink, is provided with special transmission capabilities and steady power supply in order to localize the sensors around it. In practice, the goal is to provide coordinates to the sensors with respect to the sink and not to acquire the real absolute positions. In such a context, localization is intended as the task of making each sensor able to acquire a coarse-grain location with respect to a given sink node and is referred to as *training*. The main characteristic of such training protocols relies on using a single sink node to impose a discrete polar coordinate system. The process might be executed mainly by the sink by means of asymmetric broadcasts (from the sink to the sensors) without multi-hop communications among the sensors. The sensors compute their coarse-grain location by exploiting the information received from the sink without performing any local communication. This must be intended to save energy in order to accomplish their real task of sensing. In particular, the two protocols presented in [8] assume that all the sensors are synchronized to the master clock running at the sink. Exploiting the fully synchronized model among sensors and sink and the capability of irregularly alternating between sleep and awake periods (whose frequency and length depend on the protocol computation) such protocols achieve an optimal time (in the number of coronas/sectors) for terminating the training process. Clearly, different assumptions/environments lead to different strategies. Further in the chapter will be described two new approaches [6, 9, 33], which assume that sensors and the sink are not synchronized. Moreover, also the case where sensors can exchange data in order to speed up the training process is considered in Sect. 17.4.3.

17.2.2 About the Energy Hole Problem

Concerning the energy hole problem, we classify the related work into three categories: analysis of energy hole problem, energy-efficiency design, and non-uniform sensor distribution strategies.

There has been some work on the analysis of the energy hole problem in WSNs. In [25, 26], the authors present a mathematical model to analyze the energy hole problem in circular WSNs with uniform sensor distribution and consisting of concentric coronas. They examine the validity of several possible schemes to mitigate the energy hole problem from a traffic perspective. It has been observed that in a uniformly distributed sensor network, hierarchical deployment and data compression have positive effects on the problem. Increasing data generating rate only makes it worse, whereas simply adding more sensors in the network makes little difference provided that the uniform sensor distribution is maintained. This work does not explore the possibility of *avoiding* energy hole in WSNs.

The authors in [38] are the first to analyze how to avoid the energy hole problem, assuming a WSN with uniform sensor distribution and constant data reporting. They use a common energy model (see [44]) that the energy consumed in transmitting a message of unit length is $E = d^{\alpha} + c$, where α is the energy attenuation parameter related to specific field, d is the distance between the data sender and receiver, and c is a positive system parameter. They demonstrate that if the transmission range of each sensor is adjustable, the energy spent in routing is minimized when each corona has the same width. This however would lead to unbalanced energy depletion in the network. They prove that for $\alpha > 2$, the unbalanced energy depletion is preventable, but for $\alpha = 2$, it is inevitable.

In existing protocols like LEACH [19], HEED [59], UCS [50], EECS [57], EEUC [24], and VIBE [40], hierarchical structures of the sensor network are constructed and clustering schemes are proposed for the purpose of distributing energy depletion among all the sensors. In LEACH, the cluster head rotation is used as a mechanism to balance the energy depletion, whereas both communication cost and remaining energy of sensors are considered when selecting the cluster head in HEED. Noticing that some cluster heads near or far from the sink take a heavier energy burden, schemes such as UCS, EECS, and EEUC are proposed to form clusters with different sizes. These methods produce smaller clusters in which the cluster heads consume more energy. VIBE describes a way to avoid the problem by changing the routing path to the sink nodes and by exploiting mobility aspects of the network.

In [41], two cases are summarized in which energy imbalance could appear in a network with uniform sensor distribution. In one case, with the assumption that all the sensors can communicate directly with the sink, the sensors send data to the sink via a single hop, thus the sensors farther away would consume their energy faster. In the other case, data are forwarded to the sink via multiple hops. Therefore, the sensors near the sink take more traffic loads and would drain earlier their battery. When the first such sensor uses up its energy, the network lifetime ends. The authors in [41] assume that each sensor can vary its transmission range and model the network lifetime maximization as a linear optimization problem. They state that the maximization of network lifetime can only be attained at the expense of loss in energy efficiencies. In [17], the maximization of network lifetime is formulated as a linear programming problem. The lifetime bounds of WSNs are derived with two regular topologies, namely regular linear network and regular two-dimensional network. It is shown that simply transmitting data to the nearest neighbors can achieve asymptotical near-optimal network lifetime.

Under the mixed data gathering scheme [21], an intermediate sensor either transmits data to one of its neighbors or sends directly to the sink when it forwards data, assuming that each sensor is able to communicate directly with the sink. A distributed data gathering algorithm is also proposed in [21] for evenly distributing the energy consumption among all the sensors in WSNs. It is proved that an energy-balanced mixed data gathering scheme could be better than other possible routing schemes due to the fact that the lifetime maximization, data flow maximization, and balanced energy consumption among the sensors are equivalent. However, the assumption that each sensor is within the direct reach of the sink might be infeasible. A related solution proposed in [49] models a WSN by concentric coronas, in which sensors in the same corona use the same transmission range, but different coronas have different transmission ranges. A right transmission range list is thus the decision factor for optimizing network lifetime after node deployment. The authors in [49] show that searching for the optimal transmission range list is a multi-objective optimization problem which is NP-hard and thus propose algorithms to reduce the searching complexity. Compared with existing algorithms, their solutions can make the network lifetime more than twice longer. The ideas of localized zone-based corona network division and mixed-routing strategy with data aggregation are combined in [61] to design an offline centralized solution for a transmitting data distribution which guarantees balanced energy consumption among the sensor nodes.

In order to tackle the unbalanced energy depletion and extend the network lifetime, other endeavors introduce mobility into the WSNs. In [54], the authors make use of mobile relays which stay only within two hops away from the sink to enhance the network lifetime by nearly four times. They also propose two joint mobility and routing algorithms capable of attaining the claimed results. A mobile sink is used in [30] to improve the network lifetime. The sensors near the sink would change over time with a sink moving in the network, thus mitigating the energy imbalance around the sink. It is proved that for extending the network lifetime, the best position for a static sink is the center of the circle when the WSN covers a circular area. It is further demonstrated that using a mobile sink is beneficial and the mobility trajectory should follow the periphery of the network. A joint mobility and routing scheme is also devised. The work in [30] is extended in [47] by studying the relationship between energy efficiency and load balancing. It is concluded that the optimum trajectory of the sink movement is a stationary annularity area. All sensed data are first transmitted into this area taking the shortest path and then collecting by the sink which moves along the area at an appropriate speed. Finally, energy hole healing protocol is presented in [48] using mobile sensors when energy hole appears.

Non-uniform sensor distribution strategy can be adopted to mitigate the energy hole problem. To this end, a non-uniform sensor distribution strategy is described in [27] to increase the network data capacity, defined as the amount of data received by the sink node. Additional sensors acting as pure *relays* are added to the network. A routing algorithm is also proposed in which some sensors sleep once in a while to save energy. Hence, some kind of awake–sleep scheduling is necessary.

Recently, a power-aware non-uniform sensor distribution scheme has been suggested in [28] to deal with the so-called sink-routing hole problem and aim for long-term connectivity in WSNs. Here the authors derive sensor distribution functions based on the hop counts. The sink-routing hole problem is essentially the same as the energy hole problem discussed in this chapter. However, neither [27] nor [28] addresses the possibility of avoiding the energy hole problem.

The possibility of avoiding energy hole by a non-uniform sensor distribution strategy in WSNs is discussed in [37]. Considering only energy consumption for data transmission, the authors state that balanced energy depletion can be achieved when the sensor density δ_i of the ith corona is proportional to $(k+1-i)$, where k is the optimal number of coronas. Under this scheme, sensors closer to the sink have to send data with lower rates.

In Sect. 17.5, we find that a nearly balanced energy depletion is achievable in the network if the number of sensors in coronas increases in geometric proportion as they get closer to the sink. We assume that the sensors constantly report data to the sink as in [26, 38] and we adopt a more practical energy consumption model than the one assumed in [37], which considers the energy lost in both data transmission and reception.

17.3 Our Model and Assumptions

We consider a wireless sensor and sink network to consist of one single, fixed sink, centrally placed with respect to a set of sensors uniformly and randomly deployed, as illustrated in Fig. 17.1a. The circular area containing the sensors and centered at the sink is called the *sink zone*.

It is assumed that the time is divided into slots. The sensors and the sink use equally long, in-phase slots, but they do not necessarily start counting time from the same slot. A sensor possesses three basic capabilities, sensing, computation, and wireless communication, and operates subject to the following fundamental constraints:

a. Each sensor alternates between *sleep* periods and *awake* periods—sleep–awake cycle has a total length of L time slots, out of which the sensor is in sleep mode for $L - d$ slots and in awake mode for d slots;
b. Each sensor is *asynchronous*—it wakes up for the first time according to its internal clock and is not engaging in an explicit synchronization protocol with either the sink or other sensors. We say that sensors that wake up simultaneously at time slot x are of *type* x;
c. Individual sensors are *unattended*—once deployed it is neither feasible nor practical to devote attention to individual sensors;
d. No sensor has global information about the network topology, but can hear transmissions from the sink;
e. Each sensor has a modest non-renewable energy budget and a limited transmission range r.

As shown in Fig. 17.1d, the training imposes a virtual coordinate system onto the sensor network by establishing

1. *Coronas*: The sink-zone area is divided into k coronas $C_0, C_1, \ldots, C_{k-1}$ each of fixed width $\rho > 0$. The coronas are centered at the sink and determined by k concentric circles whose radii are $\rho, 2\rho, \ldots, k\rho$, respectively;
2. *Sectors*: The sink-zone area is divided into h equiangular sectors $S_0, S_1, \ldots, S_{h-1}$, originated at the sink, each having a width of $\frac{2\pi}{h}$ radians.

For the sake of simplicity, it is assumed that all the coronas and all the sectors have the same width, although this is not strictly required. In a practical setting, the corona width might be equal to the sensor transmission range, and hence the (outer) radius r_i of corona C_i might be equal to $(i + 1)\rho = (i + 1)r$. In such a case, the corona number plus one gives the number of hops needed for a sensor-to-sink communication inside a sector. At the end of the training period each sensor acquires two coordinates: the identity of the corona in which it lies and the identity of the sector to which it belongs. In particular, a cluster is the intersection between a corona and a sector where all sensors have to acquire the same coordinates.

Once the training period has terminated, we assume a data logging application, where the sensors are required to send their sensed data constantly at a certain rate. For the sake of simplicity, we assume that each sensor generates and sends B bits of data per unit time. Sensors belonging to a corona $\{C_i | i \neq k-1\}$ will forward data generated by themselves and also by sensors from coronas $\{C_j | (i+1) \leq j \leq k-1\}$. The sensors in the outmost corona C_{k-1} need to forward only their own data. Here we do not assume data aggregation at any forwarding sensor.

The energy model in our discussion is as follows. The initial energy of each sensor is $\varepsilon > 0$, while the sink has no energy limitation. It has been observed that the communication-related energy dissipation usually dominates the total energy consumption in WSNs [41, 43]. In our model, we consider the energy loss related to both data transmission and reception. We further assume that a sensor consumes e_1 units of energy for sending one bit, while it depletes e_2 units of energy for receiving one bit, where $e_1 > e_2 > 0$. Later we will see how the constraint $e_1 > e_2$ can be relaxed. Note that the energy model in [19, 44] assumes that the energy consumption for transmitting l bits of data over a distance of d is given by $l(E_{\text{elec}} + \varepsilon d^{\alpha})$ and the corresponding energy dissipation in data reception is lE_{elec}, where E_{elec}, ε, and α are all system parameters. Thus, the values of e_1 and e_2 in our model can be computed as $E_{\text{elec}} + \varepsilon d^{\alpha}$ and E_{elec}, respectively, where d is the maximum transmission range of each sensor.

For ease of exposition, we consider the following two terms from [55]:

Definition *Balanced energy depletion* is attained in the network when sensors in all coronas use up their energy simultaneously.

Definition *Sub-balanced energy depletion* is attained in the network when sensors in all coronas *except the outmost one* exhaust their energy simultaneously.

17.3.1 Basic Modular Arithmetic

Since several derivations in this chapter employ modular arithmetic, let us offer a quick refresher of the terminology and basic results used hereafter.

Given any two integers x and m, with $m \neq 0$, let $|x|_m$ denote the *modulo* operation, that is, the non-negative remainder of the division of x by m (see [51]). Two integers x and y are *congruent modulo* m, denoted by $x \equiv y \bmod m$, if and only if $|x|_m = |y|_m$.

Let $\bullet$ indicate one of the three basic operations: *addition*, *subtraction*, and *multiplication*. The modulo operation distributes over such operations and hence $|x \bullet y|_m = ||x|_m \bullet y|_m = |x \bullet |y|_m|_m = ||x|_m \bullet |y|_m|_m$. Moreover, it is easy to prove that

Property 1 For any integers a, x, and m, with $a \neq 0$ and $m \neq 0$, $|ax|_{am} = a|x|_m$.

Let the *greatest common divisor* of integers x and y be denoted by (x, y). Therefore, letting $x = x'(x, m)$, $m = m'(x, m)$, and applying Property 1, one derives $|x|_m = (x, m)|x'|_{m'}$. It is worth noting that the division of x by y modulo m is possible only when y and m are *coprime*, i.e., when $(m, y) = 1$. Indeed, only in such a case there exists the *inverse multiplicative of* y *modulo* m, which is denoted by $\left|\frac{1}{x}\right|_m$ (as used in [51]) and is defined as that integer satisfying $x\left|\frac{1}{x}\right|_m \equiv 1 \bmod m$. The following property is widely used in this chapter:

Property 2 Given any integers x, y, z, w, and $m \neq 0$, the following holds:

1. If $ax \equiv ay \bmod m$ and $a \neq 0$ is such that $(a, m) = 1$, then $x \equiv y \bmod m$
2. If $ax \equiv ay \bmod m$ and $(a, m) = g$, then $x \equiv y \bmod m'$, where $m = m'g$

The next property shows how the values generated by the expression $|ix|_m$ vary when i assumes any integer value.

Property 3 Given two integers x and $m \neq 0$ such that $(x, m) = g$, the congruence $ix \equiv y \bmod m$ has solution for any $y = gy'$ with $y' \in [0, \ldots, m' - 1]$, where $m' = \frac{m}{g}$. Moreover, $|ix|_m$ generates only the values multiple of g in $[0, 1, \ldots, m-1]$, one for each different value of $|i|_{m'}$.

In the particular case where $(x, m) = 1$, the property above shows that, when i assumes all the m integer values in $[0, \ldots, m - 1]$, the expression $|ix|_m$ generates all the m integer values in $[0, \ldots, m - 1]$.

17.4 Localization Problem

The main goal of this section is to present the details of the basic training protocol, called Flat– [6]. Other protocols can be found in the literature for training purposes [7, 9, 33] in similar environments.

With Flat– protocol, each sensor acquires a coarse-grained localization (namely, the corona and the sector of the virtual coordinate system where it resides) just listening to the beacons of the sink and performing very simple local operations. As said, to stress the role of the sink which alone guides the sensors in the process of learning their virtual coordinates, Flat– is said to be a training protocol. Flat– consists of two phases: the corona phase and the sector one. During the corona training protocol, the sensors learn the corona where they reside and the sink broadcasts using the isotropic antenna, whose transmissions cover circles of fixed radius, centered at the sink itself. For example, Fig. 17.1b shows a sink (isotropic) broadcast in which the isotropic antenna transmits over a circle of radius R. The isotropic antenna can modulate transmissions with the discrete radii ρ, 2ρ, ..., $(k - 1)\rho$. When the sink transmits up to a distance $(c + 1)\rho$, it broadcasts beacon c. Such a beacon is heard by any awake sensor that resides up to corona c, i.e., at a distance $\leq c\rho$ from the sink, whereas the beacon c cannot be heard by any awake sensor that resides in corona $c + 1$ or in a larger one.

Similarly, during the sector training protocol, the sensors learn the sector where they reside and the sink broadcasts using the directional antenna. A sink (directional) broadcast of width α covers a portion of a circle of radius $k\rho$, with center in the sink, enclosed between two radii forming the angle of α radians (see Fig. 17.1c). The directional antenna can modulate h different discrete angles of width $(i+1)\dfrac{2\pi}{h}$, with $0 \le i \le h-1$. When the sink transmission covers an angle of width $(i+1)\dfrac{2\pi}{h}$, the sink broadcasts beacon i. Figure 17.1c illustrates a sink broadcast of width $5\dfrac{2\pi}{8}$.

During the corona (sector) protocol, each sensor saves in a register R the beacons that it hears while it is awake. At the end of protocol, if each sensor has been awake while the sink has transmitted all the possible k corona (h sector) beacons, it surely becomes *trained*, that is, it knows the corona (sector) where it resides. Specifically, the sensor resides in the corona (sector) corresponding to the smallest beacon it has heard.

In detail, the corona protocol executed by sink is described in Fig. 17.2. Immediately after deployment, the sink cyclically repeats a transmission cycle which involves k (isotropic) broadcasts at successively lower power levels. Each broadcast lasts for a slot and transmits a beacon equal to the identity of the outermost corona reached. Specifically, the sink starts out by transmitting the beacon $k-1$ with the highest power, sufficient to reach the sensors up to the outermost corona C_{k-1}; next, the sink transmits the beacon $k-2$ at a power level that can be received up to corona C_{k-2}, but not by the sensors in corona C_{k-1}. For the subsequent $k-2$ slots, the sink continues to transmit at decreasing power levels until it concludes its transmission cycle with a broadcast of beacon 0 that can be received only by the sensors in corona C_0. In general, at time slot z, with $z \ge 0$, the sink transmits the beacon $k-1-|z|_k$ at a power level sufficient to cover the distance $r_{k-|z|_k}$ and hence to reach all the sensors up to corona $C_{k-1-|z|_k}$, but not those beyond $C_{k-1-|z|_k}$. The sink transmission cycle is repeated for a given time τ_1 which is sufficient to accomplish the entire corona training protocol.

Two periods of the corona protocol performed by the sink are illustrated in Fig. 17.3.

The sector sink training task is analogous to the corona training task, except that now the sink broadcasts using the directional antenna. Indeed, the sink cyclically repeats a transmission cycle of h directional broadcasts with successively smaller angles. Specifically, at time slot z, with $z > 0$, the sink transmits the beacon $h-1-|z|_h$ which can reach all the sensors up to sector $S_{h-1-|z|_h}$, namely, using an angle of transmission $\alpha = (h-|z|_h)\frac{2\pi}{h}$.

```
Procedure Sink $(k, \tau_1)$;
    for $z := 0$ to $\tau_1 - 1$ do
        transmit the beacon $|k-1-z|_k$ up to corona $C_{|k-1-z|_k}$;
```

Fig. 17.2 The corona training Flat-protocol for the sink

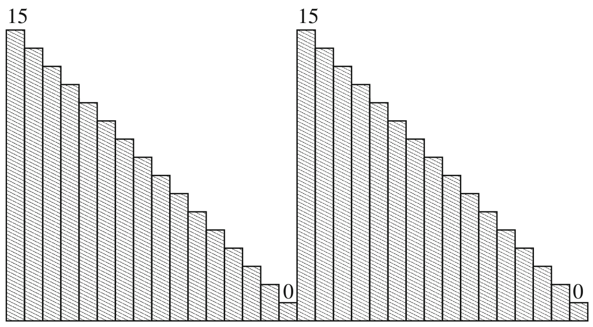


Fig. 17.3 The sink behavior when $k = 16$. A bar of height c represents a sink isotropic broadcast up to corona c, with $0 \leq c \leq k-1$

In order to describe the basic corona protocol for sensors, we assume that each sensor is aware of the sink behavior and of the total number k of coronas. In particular, k can be either stored in the sensor memory before deployment or transmitted by the sink in the beacon along with the corona identity. Immediately after deployment, each sensor wakes up at random within the 0th and the $(k-1)$th time slot and starts listening to the sink for d time slots (that is, its awake period). Then, the sensor goes back to sleep for $L - d$ time slots (that is, its sleep period). Such a sleep/awake transition will be repeated until the sensor learns the identity of the corona to which it belongs, that is, until the sensor will be trained. Each sensor, during the training process, uses a k-bit register R to keep track of the beacons, i.e., corona identities, transmitted by the sink while the sensor is awake. As soon as the sensor hears a sink transmission for the first time, it starts to fill its register R and it is able to learn the sink global time t within the current sink transmission cycle, that is, $t = |z|_k$. From now on, such a time will regularly increase so that the sensor can derive from t the beacon $|k-1-t|_k$ that the sink is transmitting. Then, in each time slot when the sensor is awake, one entry of R can always be set to either 0 or 1. In fact, if the sensor hears beacon c, then it sets $R_c = 1$, while if the sensor hears nothing, it sets $R_{|k-1-t|_k} = 0$.

Note that an awake sensor in corona C_c hears any broadcast which transmits beacon b, with $c \leq b \leq k-1$ (clearly, different beacons are heard in different broadcasts). In contrast, an awake sensor in corona C_c cannot hear the broadcasts transmitting beacon b, with $0 \leq b \leq c-1$, because it is out of the range covered by the sink transmission. Hence, if a sensor sets $R_c = 0$ (resp., $R_c = 1$), then it belongs to a corona whose identity is higher than (resp., smaller than or equal to) c. Note that only the sensors in corona C_0 can hear beacon 0 and thus they are the only ones which can set $R_0 = 1$. From the above discussions, the following *training condition* holds:

Lemma 1 [56] *A sensor which belongs to corona C_c, with $c > 0$, is trained as soon as the entries R_c and R_{c-1} of its register R are set to* 1 *and* 0*, respectively. A sensor which is in corona C_0 is trained as soon as R_0 is set to* 1.

```
   Procedure Flat– $(k, L, d)$;
1    heard := trained := false; $\nu := 0$;
2    while wakeup and $\neg$ trained do
3        $\nu := \nu + 1$;
4      for $i := 0$ to $d - 1$ do
5          if received beacon $c$ then
6              if $\neg$ heard then
7                  heard := true, $t := k - 1 - c$;
8              $R_c := 1$;
9              if $c = 0$ or ($R_c = 1$ and $R_{c-1} = 0$) then
10                 mycorona := $c$, trained := true;
11             $t := t + 1$;
12         else
14             if heard then
15                 $c := k - 1 - |t|_k$;
16                 $R_c := 0$;
17                 if $R_{c+1} = 1$ then
18                     mycorona := $c$, trained := true;
19                 $t := t + 1$;
20     if heard then
21         alarm-clock := $t := t + L - d$;
22     else
23         alarm-clock := alarm-clock + $L$;
24     go to sleep until the alarm-clock rings;
```

Fig. 17.4 The Flat– corona protocol for a sensor

In the resulting Flat– sensor protocol (Fig. 17.4), each sensor counts in ν the number of its sleep/awake transitions needed to be trained (line 1),[1] initializing its local time when the sensor receives a beacon for the first time from the sink (that is, when *heard* is set to `true` in line 7) and stores in *alarm-clock* the time when the next sleep/awake transition is planned (lines 21–23). It is worthy to note that after having set *heard* to true, if a sensor is awake and does not hear the beacon, since it knows the beacon broadcast by the sink, it can set the corresponding register entry to 0 (lines 14–16). After any entry of R is filled, the sensor checks the training condition stated in Lemma 1. In the procedure, each sensor executes $O(1)$ arithmetic/logic operations per time slot.

When $k = 12$, $L = 16$, and $d = 4$, a sensor of type $x = 5$ (i.e., it wakes up for the first time at time slot 5) belonging to corona $c = 3$ fills the register R during the Flat– protocol as reported in Fig. 17.5 (where register R is depicted as a circular register). The sensor wakes up for the first time while the sink transmits $|k - 1 - x|_{12} = 6$ and fills $R_6 = 1$. During the same awake period, also $R_5 = R_4 = 1 = R_3 = 1$ are filled. In the first slot of the second awake period (i.e., $i = 1$), the sensor sets $R_2 = 0$ because it cannot hear the beacon, but it knows that the sink is transmitting $|k - 1 - (x + L)|_k = |6 - 16|_{12} = 2$. At this point, the sensor becomes trained, but

[1] Since the behavior of the sensor does not depend on which awake period it is, ν could be omitted from the algorithm description. Indeed this is required just for the analysis purpose.

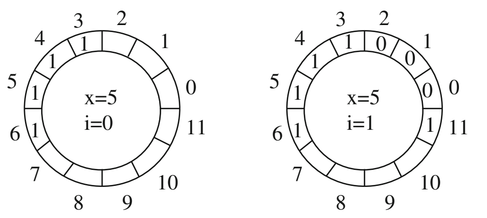


Fig. 17.5 Status of the register R of a sensor of type $x = 5$ belonging to corona $c = 3$ when $L = 16$, $k = 12$, and $d = 4$ at the ith awake period, with $0 \leq i \leq 1$

it continues to fill the register R, setting $R_1 = R_0 = 0$ and $R_{11} = 1$ until it goes to sleep. Note that the sensor becomes trained during the awake period $i = 1$ and ignores further sink transmissions.

Sector training is the same as corona training also for the sensor behavior, once one has replaced h with k, *mysector* with *mycorona*. Clearly, a similar sector training condition can be stated:

Lemma 2 *A sensor which belongs to sector S_c, with $0 < c \leq h - 1$, is trained as soon as the entries R_c and R_{c-1} of its register R are set to 1 and 0, respectively. A sensor which is in sector S_0 is trained as soon as R_0 is set to 1.*

Thus, sector training protocol for sensors will not be further discussed and all the results that follow will be presented for coronas only.

17.4.1 Correctness and Performance Analysis

The challenge now is to derive conditions on k, L, and d which guarantee that all the sensors are trained, independent of their type and of the corona c they belong to.

By the training conditions discussed above, a sensor could not be trained if it does not receive at least the beacon of the corona where it resides. Then, it is sufficient to guarantee that, while a sensor is awake, all the beacons are transmitted by the sink. This is trivially true if $d \geq k$. However, when $d < k$, let us look on special values of L, k, and d for which the training protocol succeeds for all sensors. Let $|L|_k = d$. Consider now two consecutive awake periods, say i and $i+1$, for a sensor of type x, $0 \leq x \leq k - 1$. The ith period covers the time slots $x + iL, x + iL + 1, \ldots, x + iL + (d - 1)$, while the $(i+1)$th period covers the slots $x + (i+1)L, x + (i + 1)L + 1, \ldots, x + (i + 1)L + (d - 1)$. During two such awake periods, the sink transmits the beacons $|x + iL|_k, |x + iL + 1|_k, \ldots, |x + iL + (d - 1)|_k$ and $|x + (i + 1)L|_k, |x + (i + 1)L + 1|_k, \ldots, |x + (i + 1)L + (d - 1)|_k$, respectively, which are consecutive. Indeed, by the modular arithmetic properties refreshed in Sect. 17.3.1, during the last time slot of the ith period the sink transmits beacon $|x + iL + (d - 1)|_k = |x + id + (d - 1)|_k$, while during the first time slot of the $(i+1)$th period it transmits beacon $|x+(i+1)L|_k = |x+id+d|_k$. Although the time slots covered by the ith and the $(i + 1)$th awake periods are not consecutive in the global time, the beacons transmitted by the sink are consecutive. In other words, the sensor receives the sink transmissions as it was awake without interruptions. Thus, regardless of the sensor type, when $|L|_k = d$, a sensor requires $\lceil \frac{k}{d} \rceil$ consecutive

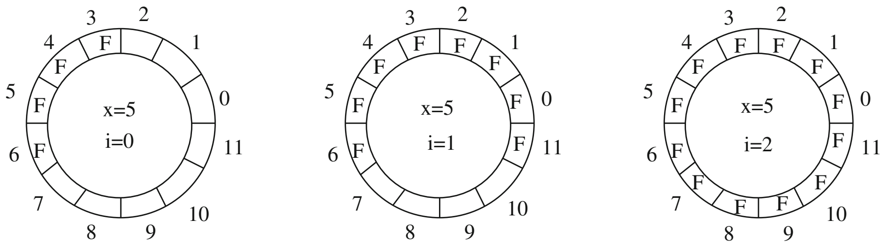


Fig. 17.6 Status of the register R of a sensor of type $x = 5$ when $L = 16$, $k = 12$, and $d = 4$ at the ith awake period, with $0 \leq i \leq 2$. Initially R is empty. When the sink transmits beacon c, R_c is filled

sleep–awake cycles to become trained because during such a period all the corona beacons are transmitted by the sink.

Figure 17.6 shows how a sensor of type $x = 5$ fills its register R during the Flat–protocol when $L = 16$, $k = 12$, and $d = |L|_k = |16|_{12} = 4$. Since we are just interested in the positions of R which are filled, we mark R_c with F, independent of the value 0 or 1 saved in R.

For arbitrary values of L and k, since any two consecutive awake periods differ by L time slots, they start when the sink is transmitting two corona beacons that differ, by Property 1, by $(L, k)|L'|_{k'} \geq (L, k)$, $k' = \frac{k}{(L,k)}$ and $L' = \frac{L}{(L,k)}$. Thus, the beacon heard at the wakeup time repeats every k' awake periods, and the process is periodic. In each awake period, the sensor is awake while the sink transmits d consecutive beacons. Clearly, if $d < (L, k)$,[2] the sensor cannot be awake during the transmissions of all corona beacons, independent of how long it stays awake. As an example, consider Fig. 17.7 where $x = 5$, $L = 16$, $k = 12$, and $d = 3$. When the sensor wakes up in two consecutive awake periods, the sink is transmitting two beacons whose values differ by 4 because $k' = 3$, $L' = 4$, and $(L, k)|L'|_{k'} = 4$. Thus, since $k' = 3$, even if the Flat– protocol runs longer, the register R cannot be entirely filled. Therefore, if $d < (L, k)$, there are sensors that cannot hear the

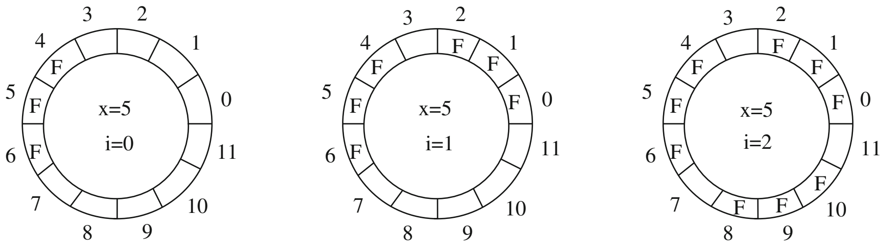


Fig. 17.7 Status of the register R of a sensor of type $x = 5$, when $L = 16$, $k = 12$, and $d = 3$ at the ith awake period, with $0 \leq i \leq 2$. The empty positions R_3, R_{11}, and R_7 will remain unfilled independent of how long the protocol runs

[2] Recall that (L, k) denotes the greatest common divisor between L and k.

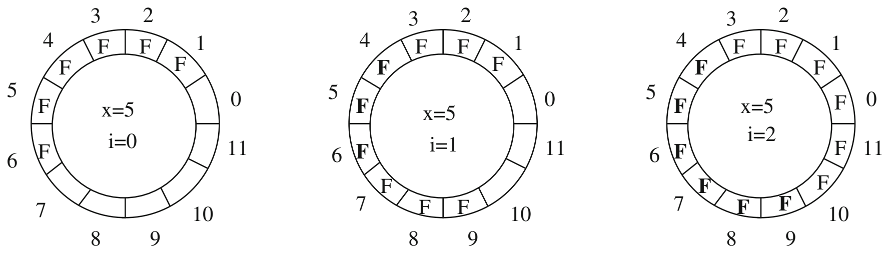


Fig. 17.8 Status of the register R of a sensor of type $x = 5$, when $L = 21$, $k = 12$, and $d = 6$ at the ith awake period, with $0 \leq i \leq 2$. The positions filled more than once are depicted in *bold*. Note that $(L,k) = 3$ and the beacons transmitted first by the sink in each awake period differ by a multiple of 3

beacon of the corona where they reside and then remain untrained. In other words, $d \geq (L,k)$ is a necessary condition for training all the sensors.

Whereas, if $d \geq (L,k)$, each sensor completely fills register R. In this case, each sensor can hear the same beacon more than once and R can be filled in a not consecutive manner. However, each sensor is trained after at most k' awake periods. In the example in Fig. 17.8, the first awake period starts while the sink transmits beacon $|k-1-x|_k = |11-5|_{12} = 6$, the second period while beacon $|k-1-(x+L)|_k = |11-(5+21)|_{12} = 9$ is transmitted, and the third while beacon $|k-1-(x+2*L)|_k = |11-(5+42)|_{12} = 0$ is transmitted. Although the process starts repeating after the fourth sleep–awake cycle because $k' = \frac{12}{3} = 4$, the register R is filled already at the end of the third cycle since $d > (L,k) = 3$.

The previous discussion has been formalized in [6]. Let us survey here the formal results that we obtained.

Lemma 3 *(Barsi et al.* [6]*) Given L, d, and k, there are exactly $k' = \frac{k}{(L,k)}$ different beacons (or corona identities) that can be transmitted by the sink when the sensor starts any awake period. Consider a sensor of type x, $0 \leq x \leq k-1$. The beacon transmitted when the sensor starts its ith awake period is given by $|K_x - iL|_k = \big|K_x - i(L,k)|L'|_{k'}\big|_k$, where $K_x = C_{|k-1-x|_k}$ is the beacon transmitted at time x. Overall only k' different beacons can be transmitted by the sink when the sensor starts its awake periods, independent of how long the training process will be. Such k' beacons can be reindexed as $|K_x - s(L,k)|_k$, for $0 \leq s \leq k'-1$.*

Obviously, the process is repeated every k' sleep–awake cycles. At the beginning of the k'th awake period, the sensor and the sink are in the same reciprocal state as they were at the beginning of the 0th one, with the only difference that the sensor has heard the sink at least once. Thus, in at most k' further sleep–awake cycles, the sensor can be trained. It follows:

Lemma 4 *(Barsi et al.* [6]*) Given L, d, and k, all the entries of R that the sensor can fill are set within the first $\frac{2k}{(L,k)}$ sleep–awake cycles. Moreover, all the sensors are trained if and only if $d \geq (L,k)$.*

Recalling that $\left|\frac{1}{L'}\right|_{k'}$ denotes the inverse multiplicative of L' mod k', that is, the smallest integer μ such that $|L'\mu|_{k'} = 1$, the following two lemmas determine, for $d = (L, k)$ and $d = |L|_k$, respectively, exactly in which awake period $i_{c,x}$ a sensor of type x is awake while the sink transmits a given beacon c.

Lemma 5 *(Barsi et al.* [6]*) Let c be any corona identity and assume $d = (L, k)$. The sink transmits the beacon c during the $i_{c,x}$th awake period of a sensor of type x, where $i_{c,x} = \left|\left\lfloor \frac{|K_x - c|_k}{d} \right\rfloor \left|\frac{1}{L'}\right|_{k'}\right|_{k'}$, $L' = \frac{L}{d}$, and $k' = \frac{k}{d}$.*

Lemma 6 *(Barsi et al.* [6]*) Let c be any corona identity and assume $d = |L|_k$. The sink transmits beacon c during the $i_{c,x}$th awake period of a sensor of type x, where $i_{c,x} = \left\lfloor \frac{|K_x - c|_k}{d} \right\rfloor$.*

Since a sensor belonging to corona c is trained when it has filled both R_c and R_{c-1}, it yields:

Lemma 7 *(Barsi et al.* [6]*) Let $d = (L, k)$. A sensor of type x which belongs to corona c, with $c > 0$, is trained during the ith awake period where $i = i_{c-1,x}$, if $i_{c,x} \le i_{c-1,x}$, or $i \le i_{c,x} + \left|\frac{1}{L'}\right|_{k'}$, if $i_{c,x} > i_{c-1,x}$. If $c = 0$, then $i = i_{0,x}$.*

Lemma 8 *(Barsi et al.* [6]*) Let $d = |L|_k$. A sensor of type x which belongs to corona c, with $c > 0$, is trained during the ith awake period where $i = i_{c-1,x}$, if $i_{c,x} \le i_{c-1,x}$, or $i \le i_{c,x} + 1$, if $i_{c,x} > i_{c-1,x}$. If $c = 0$, then $i = i_{0,x}$.*

From Lemmas 7 and 8, one knows the number of sleep–awake periods required by any sensor in the network to be trained.

Since the protocol terminates when all the sensors are trained, we measure the protocol performance by counting the maximum number of transitions, say $\nu_{\max}$, required by any sensor to be trained.

Thus, the worst-case performance for the Flat– protocol is stated as follows:

Theorem 1 *(Barsi et al.* [6]*) Given L, d, and k, if $d < (L, k)$ then there are sensors that cannot be trained by the Flat– protocol; otherwise all the sensors are trained, and*

1. *If $(L, k) \le d < |L|_k$, then $\nu_{\max} \le \dfrac{k}{(L, k)} + \left|\dfrac{1}{L'}\right|_{k'}$, where $k' = \dfrac{k}{(L, k)}$ and $L' = \dfrac{L}{(L, k)}$;*
2. *If $|L|_k \le d < k$, then $\nu_{\max} \le \left\lfloor \dfrac{k}{|L|_k} \right\rfloor + 1$;*
3. *If $d = k$, then $\nu_{\max} = 2$.*

We now turn to the analysis of the average case performance of the Flat– protocol, where it is assumed that the type x is a discrete random variable uniformly distributed in $[0, k-1]$. Let N be the total number of sensors, let N_c be the number of sensors that belong to corona c and, among them, let $N_{c,x}$ be those of type x, with $0 \leq c, x \leq k-1$. Since x is uniformly distributed, $N_{c,x} = \frac{N_c}{k}$ and, clearly, $\sum_{c=0}^{k-1} N_c = N$. Letting ν_{avg} and ω_{avg} denote the average number of transitions and the average overall sensor awake time, respectively, one has the following result.

Theorem 2 *(Barsi et al.* [6]*) Given L, d, and k, if $d < (L,k)$ then there are sensors which cannot be trained by the Flat– protocol; otherwise all the sensors are trained, and:*

1. *If $(L,k) \leq d < |L|_k$, then $\nu_{\text{avg}} \leq \frac{k'+1}{2} + \frac{1}{(L,k)} \left|\frac{1}{\overline{L'}}\right|_{k'}$, where $k' = \frac{k}{(L,k)}$ and $L' = \frac{L}{(L,k)}$;*
2. *If $|L|_k \leq d < k$, then $\nu_{\text{avg}} \leq \left(\left\lfloor \frac{k}{|L|_k} \right\rfloor + 1\right)\left(\frac{1}{2} + \frac{|k|_{|L|_k} + 1}{k}\right) - \frac{1}{k}$;*
3. *If $d = k$, then $\nu_{\text{avg}} \leq 1 + \frac{1}{k}$.*

It is worthy to point out that, as shown by Theorems 1 and 2, the performance of the Flat– protocol depends on the values of k, d, and L. Therefore, it is conceivable that in mission critical systems these parameters could be tuned before the sensor deployment in order to guarantee a predefined performance quality.

17.4.2 Improvements

By exploiting two basic observations, some improvements to the Flat– protocol are proposed below. First of all, notice that as soon as a sensor hears the sink transmission for the first time, it learns from the beacon the sink global time modulo the sink transmission cycle. Therefore, it can immediately retrieve backward the beacons which it did not hear and which were transmitted by the sink during its previous awake periods, setting to 0 the corresponding entries of R. This results in the so-called Flat protocol. Second, when a sensor hears a beacon c, it knows that it will also hear all the beacons greater than c, and thus it can immediately set to 1 the entries from R_c up to R_{k-1}. Similarly, when a sensor sets an entry R_c to 0, it knows that it cannot hear any beacon smaller than c, and thus it can immediately set to 0 the entries from R_{c-1} down to R_0. In contrast to the original protocol, the sensor now fills entries of R relative to beacons not yet transmitted during its awake periods. Therefore, it can look ahead to decide whether it is worthy or not to wake up in the next awake period. If the d entries of R that will be transmitted by the sink in the next awake period have already been filled, then the sensor can skip its next awake period, thus saving energy. The sensor repeats the look-ahead process above until it finds a future awake period whose corresponding d entries are not already filled. The resulting protocol, called Flat+, leads to the following results:

Theorem 3 *(Barsi et al.* [6]*) Given L, d, and k, if $d < (L,k)$ then there are sensors which cannot be trained by the Flat+ protocol; otherwise all the sensors are trained, and*

1. *If* $(L,k) \leq d < |L|_k$*, then* $\nu_{\max} \leq \dfrac{k}{(L,k)}$*;*
2. *If* $|L|_k \leq d < k$*, then* $\nu_{\max} \leq \left\lceil \dfrac{k}{|L|_k} \right\rceil$*;*
3. *If* $d = k$*, then* $\nu_{\max} = 1$.

17.4.3 The Cooperative Protocol

Another interesting approach used to accomplish the training procedure is the so-called *Cooperative* protocol (*Coop*, for short) [33]. The key idea here is to train only a subset of sensors, uniformly spread over all the coronas composing the network, so that the time while the sink transmits is reduced. Then, by means of local and cheap transmissions, the trained sensors propagate the acquired coordinates to the large number of remaining untrained sensors which share the same coordinates. Finally, exploiting the periodicity of the awake–sleep cycle, the sensors become trained independent of their type. So, one further assumption with respect to the model described in Sect. 17.3 is that during an awake period, sensors can listen to the sink (as usual) or listen to the surrounding sensors or transmit to the sensor neighbors. Moreover, a sensor can perform transmissions in two modalities by switching the transmission range to $r = \rho$ for routing purposes or to $r < \frac{\rho}{2}$ for the cooperative training protocol. If an awake sensor receives more than one message at the same time, we assume that it correctly receives the message only if all the transmissions refer to the same message. Otherwise, the sensor hears noise. The rationale behind the latter assumption comes from a common engineering technique used in today's commodities. To boost the communication range, for example, wireless LAN access points use several antennas connected via hardware that transmit simultaneously. Not only is the packet received correctly, but also it can be transmitted further. The same idea has already been adopted to strengthen the sensor radio communication (see for instance [32]). From now on, we assume $d \geq 2$.

The cooperative training consists of three stages. The first stage is deterministic and it trains sensors independent of the network density, whereas the success of the other two stages strictly depends on the network density.

From now on, we will assume the awake–sleep period $L = k$ and the corona width $\rho = 1$. In the first stage, which starts immediately after the deployment or the movement of the sink, the Flat$-$ protocol, described in the previous section, is performed for only $k+d-1$ time slots (Fig. 17.9). Thus, at the end of the first stage, a subset of sensors, called *seeds*, has been trained directly by the sink. Hence, the localization acquired by the seeds is always correct. Although the seeds in different coronas have different types, in each corona we have almost the same density of seeds because there are $d-1$ types which are seeds in each corona and the sensors of the same type are uniformly distributed in the network.

In the second stage, the seed sensors boost the cooperative process. When the seeds wake up, they broadcast the corona identity just learnt in their transmission area of range $r < \rho/2$. The awake untrained sensors are in the receiving mode. If an

```
Procedure Sink $(k, d)$;
    for $t := 0$ to $k + d - 2$ do
        transmit the beacon $|k - 1 - t|_k$ up to corona $C_{|k-1-t|_k}$;
```

Fig. 17.9 The corona training Cooperative protocol for the sink

untrained sensor receives a corona identity from one sensor or concordant corona identities in the same time slot from more than one sensor, it becomes trained and starts to broadcast at its turn. In contrast, if an untrained sensor receives different corona identities because it is in the transmission area of two trained sensors belonging to two different coronas, it hears noise. Such a sensor becomes a *white-flag* sensor, and it has to wait the third stage to acquire its localization. The second stage proceeds for $2k$ time slots. Note that, at this stage the sink does not participate. In fact, at this point sensors are completely independent of the sink which might have been already moving for covering other zones.

The third stage involves all the white-flag sensors and only the trained sensors that belong to the coronas with even identities. This stage lasts for one awake–sleep cycle. Specifically, the trained sensors in the coronas with even identities broadcast their corona identity, while the awake white-flag sensors listen in order to acquire their localization. Clearly, in this way, a white-flag sensor can only learn an even corona identity, thus acquiring an approximate localization.

The sensor protocol is shown in Fig. 17.10.

17.4.3.1 Analysis of the Coop Protocol

In order to analyze which sensors become seeds in the first stage in each corona, let us observe that the sensors of type x receive the same beacons independent of the corona to which they belong, but they behave differently from one corona to another. In fact:

Lemma 9 *(Navarra et al.* [33]*) The seed in corona γ, $1 \leq \gamma \leq k-1$, is the sensors of type $x = |k - 1 - \gamma - w|_k$ with $w = [0, d - 2]$, or equivalently:*

$x \in \left[|k - \gamma - d + 1|_k, |k - 1 - \gamma|_k\right]$ if $|k - \gamma - d + 1|_k \leq |k - 1 - \gamma|_k$, $x \in \left[|k - \gamma - d + 1|_k, k - 1\right] \cup \left[0, |k - 1 - \gamma|_k\right]$ if $|k - \gamma - d + 1|_k > |k - 1 - \gamma|_k$. Similarly, the seeds in corona 0 are those with type $x = |k - 1 - w|_k$ with $w = [0, d - 1]$, or equivalently $x \in [|k - d|_k, |k - 1|_k]$.

The sensors enter in their second stage at a global time $t \geq k + d - 1$, when the sink does not broadcast anymore. Specifically, the sensors of type $0 \leq x \leq d - 2$ enter in the second stage at their third awake period and thus they remain inactive during their second awake period. Whereas the sensors of type $d - 1 \leq x \leq k - 1$ enter in the second stage at their second awake period.

The second stage lasts $2k$ time slots, starting from the global time slot $k + d - 1$. Recalling that a sensor of type x wakes up for the ith awake period, with $i \geq 1$, at time slot $x + (i - 1)L$, and that $L = k$, in the interval $t \in [k + d - 1, 2k + d - 2]$, all the types of sensors enter in the second stage. In fact, at time t, the sensors of type $x = |t|_L = |t|_k$ wake up. Thus, during the interval $t \in [k + d - 1, 2k - 1]$ the

```
Procedure Sensor
     Input: $x, d, L, k, r, j$;
1.   case $j$ :
       $j = 1$ :
2.     for $t := 0$ to $d - 1$                                   {Initialize}
3.       $C[t] := -1$;
4.     trained := white-flag := seed := false;
5.     corona := $-\infty$; $\tau := -1$;
6.     for $t := 0$ to $d - 1$                                   {First stage}
7.       $C[t]$ := listen-sink($\gamma$);
8.       if $C[t] = 0$
9.       then trained:= seed := true; corona:= 0;
10.      else if ($t \geq 1$ and $C[t] = \emptyset$ and $C[t-1] = \gamma$)
11.          then trained:= seed:=true, corona:= $C[t-1]$;
12.      $\tau := \tau + 1$;
13.    if $x \geq d - 2$
14.    then $j := 2$; set-alarm-clock($\tau + L - d$);
15.    else $j := 3$; set-alarm-clock($\tau + 2L - d$);
       $j = 2, 3$ :
16.    for $i := j$ to 3                                         {Second stage}
17.      for $t := 0$ to $d - 1$
18.        if trained
19.        then broadcast(corona)
20.        else if $\neg$ white-flag
21.            then listen-sensor(corona);
22.              if corona $\neq \emptyset$
23.              then corona := compatible(corona);
24.                   trained := true;
25.              if corona =noise then white-flag:=true;
26.      $\tau := \tau + 1$;
27.      if (white-flag or (trained and $\neg$ seed) or
           (seed and $x \notin [d-1, |2d-3|_k]$))
28.      then $j := 4$; set-alarm-clock ($\tau + (3-i)L + L - d$);
29.      else $j := j + 1$; set-alarm-clock ($\tau + L - d$);
       $j = 4$ :
30.    for $t := 0$ to $d - 1$                                   {Third stage}
31.      if trained and $|\text{corona}|_2 = 0$
32.      then broadcast(corona)
33.      else if white-flag
34.          then listen-sensor(corona);
35.            if corona $\neq \emptyset$ then trained:=true;
36.      $\tau := \tau + 1$;
37.    set-alarm-clock($\tau + L - d$);
```

Fig. 17.10 The corona training protocol for the sensor

sensors of type $x \in [d-1, k-1]$ wake up because they enter in the second stage in their second awake period, while during the period $t \in [2k, 2k+d-2]$ those of type $x \in [0, d-2]$ wake up because they enter in the second stage during their third awake period. Moreover:

Lemma 10 *(Navarr et al. [33])* In the interval $t \in [k+2d-3, 2k+2d-3]$, all the sensors of the $d-1$ types $|t-w|_k$, with $w = [0, d-2]$, are awake simultaneously.

The cooperative process becomes effective in each corona when all the seeds are awake and broadcast. Thus, this happens for the first time, by Lemma 9, in corona $\gamma = |k-2d+2|_k$ at time slot $k+2d-3$. Since $d \geq 2$, i.e., the seeds are awake simultaneously for two time slots, the seeds in corona $\gamma = |k-2d+2|_k$ are awake simultaneously and broadcast also at time slot $k+2d-2$ (lines 18–19). At that time, the sensors of the type $|2d-2|_k$, which are untrained in corona $|k-2d+2|_k$, wake up and listening to their seed neighbors, they become trained (lines 22–23). The new trained sensors start to broadcast for the remaining $d-1$ time slots of their awake period, replacing the seed of type $|t-d+1|_k = d-1$ that go back to sleep. This is repeated for $k-d$ time slots up to time $2k+2d-3$, training all the types of sensors in corona $|k-2d+2|_k$.

The above reasoning can be generalized as follows:

Theorem 4 *(Navarr et al. [33])* The cooperative training process is effective during the time slot $k+2d-2+y$, with $0 \leq y \leq k-1$ in corona $\gamma = |k-2d+2-y|_k$, and in such a corona, all the types of sensors are trained at time $2k+d-2+y$, with $0 \leq y \leq k-1$.

Observe that the last corona to be trained is corona $|k-2d+3|_k$ where the process lasts from time $2k+2d-3$ up to $3k+d-3$. Moreover, note that at time slot $3k+d-3$ the sensors of type $d-2$, which entered as last in the second stage, have just completed their third sleep–awake cycle.

Up to now, it has been assumed that during the second stage each untrained sensor receives concordant and correct corona identities. Nonetheless, since all the sensors of the same type are always awake simultaneously independent of the corona to which they belong, it may happen that sensors of the same type have learnt different corona identities in different coronas and broadcast discordant coronas or may have different status (i.e., seed, untrained, trained) and they can produce errors. Consider, for example, the sensors of type $\overline{x} = |k-\gamma-d+1|_k$, with $0 \leq \gamma \leq k-1$ during the second stage. When such sensors wake up, they start to broadcast in corona γ where they are seed, whereas they listen in corona $\gamma-1$ where they are untrained. A sensor of type $\overline{x}$ on the border of corona $\gamma-1$ can receive only the corona identity γ and thus it could acquire a wrong localization. In this case, however, the correct localization can still be derived by the beacon received in the first stage:

Lemma 11 *(Navarr et al. [33])* If a sensor receives the corona identity γ and it has already heard beacon $\gamma-1$ in the first stage, it learns to belong to corona $\gamma-1$.

Unfortunately, not always the error can be repaired. Thus, at the end of second stage, on the border of the coronas there are white-flag sensors as well as trained sensors whose localization is not correct. However, if the density of the trained sensors is sufficiently high, mainly white-flags are in the corona borders. During the third phase, then, the white-flag sensors in the coronas with even identities are turned into trained sensors, while those in the coronas with odd identities acquire a corona identity which differs at most ± 1 from the correct one.

So far we have not paid attention to the untrained sensors, that is, those sensors that, during the entire training process, cannot hear the transmissions of the sink or of any trained sensor. However, by exploiting the density of the network and stochastic arguments, with high probability, there are no untrained sensors when

$$\frac{N}{k} \geq (1+\varepsilon)\log\left(\frac{N}{k}\right)\frac{k^2}{r^2} \tag{17.1}$$

for any fixed $\varepsilon > 0$.

17.4.4 Experimental Results

In [33], the Flat and the Coop protocols have been compared with respect to the quality of the achieved localization and the power consumption per sensor. In the simulation, each corona has a unit width and N sensors are uniformly distributed within a circle of radius k, centered at the sink. Moreover, each sensor independently generates its type x, as an integer, selected at random and uniformly distributed in the range $[0, k-1]$.

By varying among the three parameters N, k, and r, we consider two different settings, as shown in Table 17.1. Moreover, let $\mathbb{E}(N_s) = \mathcal{O}\left(\frac{N}{k}\right)$ be the expected number of sensors of the same type $x \in [0, k-1]$ and $\lambda_s = \mathcal{O}\left(\frac{Nr^2}{k^3}\right)$ be the number of sensors for each type in a small disk of radius r. Table 17.1 reports also $\mathbb{E}(N_s)$, λ_s, and $\log(N_s)$ for the settings used in the experiments.

Clearly, both the settings simulate massive sensor networks: for example, assuming the corona width $\rho = 100$ m, there are 0.15 and 0.025 sensors per square meter in S_1 and S_2, respectively. At the present state of the technology, small sensors, supporting communications in a range varying from 10 to 100 m, like TinyNode 584 produced by Shockfish S.A. or T-node developed by SOWNet Technologies can be used for building such massive networks [1, 12].

The Flat and Coop protocols assume the same parameter values, except that Flat uses an awake–sleep cycle of length $L = k + 1$ instead of k. Indeed, when $L = k$, Flat cannot complete the training process [6], and thus the smallest value of L acceptable for Flat has been used.

Concerning the power consumptions, Table 17.2 reports the values, measured in the field, of a T-node in different operational modes [1] to have a realistic setting.

For each protocol, the maximum $p_{\max}$ (minimum $p_{\min}$, resp.) power consumed by each sensor along with the average p_{avg} power, obtained by summing up over

Table 17.1 Experimental settings

	N	k	r	$\mathbb{E}(N_s)$	λ_s	$\log(N_s)$
S_1	700000	12	$\frac{1}{7}$	58333	8.15	10.97
S_2	819200	32	$\frac{1}{4}$	25600	9.56	10.15

Table 17.2 Estimate of sensor power consumption in different operational modes with a sensor transmission range r of 20 m

Sensor mode	Power consumption
μC sleep with timer on	60 μW
μC switch on, radio startup	30 mW
μC switch off, radio shutdown	30 mW
μC active, radio idle listening	60 mW
μC active, radio TX	80 mW

all the sensor power consumptions and dividing by the number of sensors in the experiment, has been measured.

One can note that although Coop and Flat consume overall almost the same power as shown in Fig. 17.11, the difference between the sensor maximum and minimum power consumption in the Coop protocol is much less than that measured for Flat. In other words, the Coop training protocol drains the sensors in a balanced way, and therefore it works in favor of the network life span.

When k increases, like in Fig. 17.11 scenario S_2, the power effectiveness of the Coop protocol is neat. The power $p_{\max}$ of Coop is smaller than or equal to the p_{avg} of the Flat protocol for any value of d.

It is worthy to note, however, that the Flat protocol correctly trains all the sensors while the Coop protocol admits a very small percentage of mistrained sensors.

17.5 Energy Hole Problem

In this section, we analyze the non-uniform sensor distribution strategy from an energy perspective. Let us recall that we consider the following two terms from [55]:

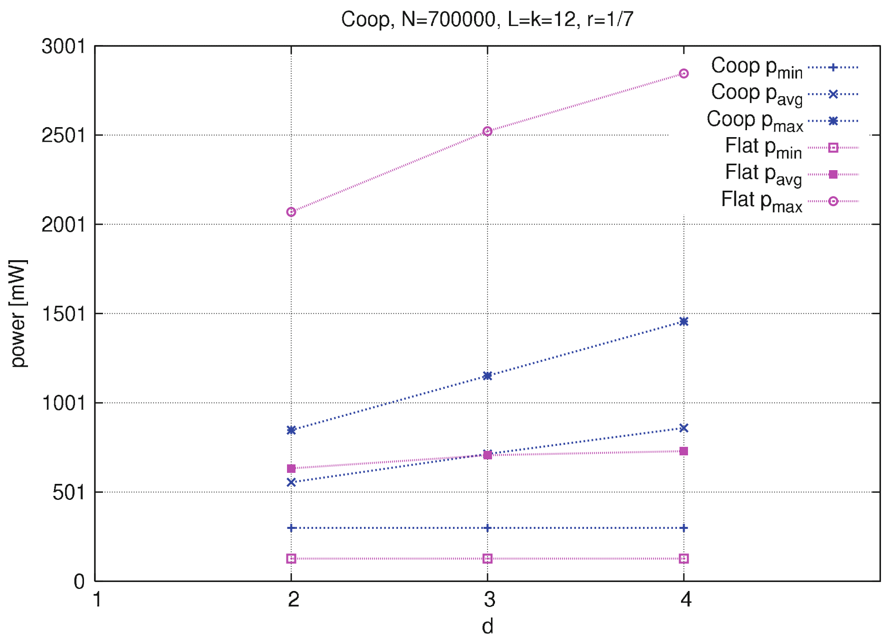
Definition *Balanced energy depletion* is attained in the network, when sensors in all coronas use up their energy simultaneously.

Definition *Sub-balanced energy depletion* is attained in the network, when sensors in all coronas *except the outmost one* exhaust their energy simultaneously.

In the following, we demonstrate that the sub-balanced energy depletion in the network is possible if the sensors have a non-uniform geographic distribution. Note that if the minimum density of the network is sufficiently high to satisfy the density requirements of the Coop protocol, both the localization algorithms presented in the previous sections work under this assumption.

17.5.1 General Non-uniform Sensor Distribution Strategy

Adding more sensors to the traffic-intensive areas is a natural way to mitigate the energy hole problem and also a salient feature of the general non-uniform sensor distribution strategy. The basic idea is that the nearer the corona is to the sink, the



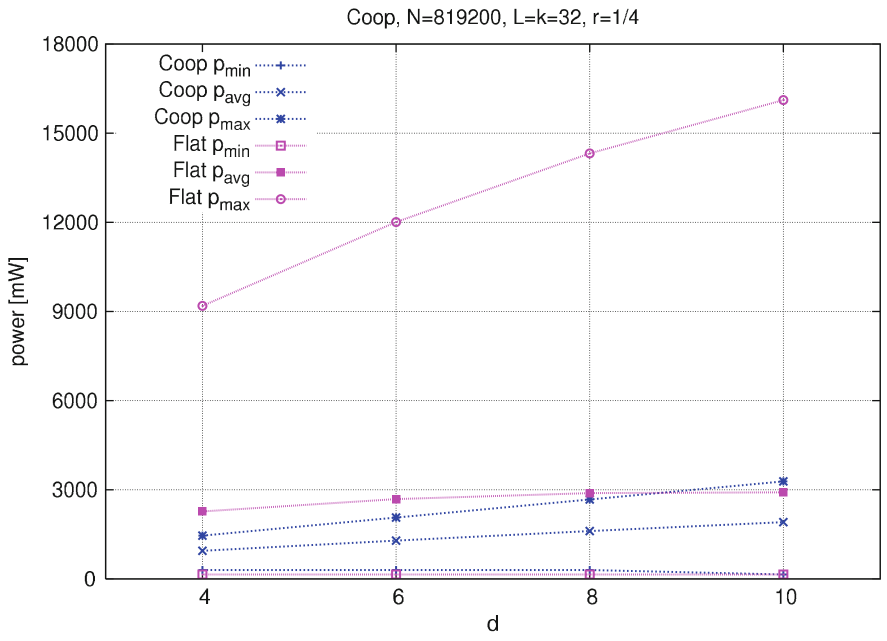


Fig. 17.11 Power consumption per sensor during the Coop protocol on setting S_1 and S_2, respectively

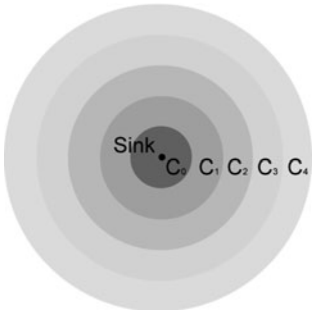


Fig. 17.12 A circular area consisting of five coronas

higher is its sensor density. Recall that in our network model, sensors belonging to inner coronas not only transmit data sensed by themselves but also forward data generated by outer coronas. As a result, these sensors deplete their energy much faster than their counterparts farther away from the sink. Hence we assign more sensors to the inner coronas of the network. Different numbers of sensors are deployed in different coronas, depending on their distance to the sink. Let us assume that sensors in the corona C_i are distributed with a density of δ_i, for $0 \leq i < k$. The sensor's density increases from the outermost corona C_{k-1} to the innermost one C_0. From the viewpoint of the whole network, the nodes are distributed non-uniformly. Therefore,

$$\delta_0 > \delta_2 > \delta_3 > \cdots > \delta_{k-1} \tag{17.2}$$

In Fig. 17.12, darker corona shows higher sensor density.

17.5.2 Energy Depletion Analysis

Let N_i and E_i, respectively, denote the number of sensors in corona C_i and the energy consumed per unit time by sensors in corona C_i. It is obvious that $N_i > N_j$ and $E_i > E_j$ when $i < j$ by non-uniform sensor distribution strategy. Data can be transmitted to the next inner corona via one hop and to the sink via i hops since we assume the width of each corona is 1 unit length, equal to the maximum transmission range of each sensor. We assume that there always exists a shortest path of i hops starting from itself and ending at the sink for each sensor in C_i, but not necessarily the same path. We show this afterward, when describing the proposed non-uniform sensor distribution strategy. Then we can calculate the total energy consumed by each corona.

Sensors in the outermost corona C_{k-1} only need to forward data generated by themselves according to the above assumptions and network model. Recalling that sensors are assumed to transmit B bits of data per unit time consuming e_1 units of energy per bit, the energy consumed per unit time by this corona is given by

$$E_{k-1} = N_{k-1} e_1 B$$

In contrast, all the sensors in other coronas have to transmit data generated by themselves as well as data originated from outer coronas. It follows that

$$E_i = B\left[N_i e_1 + \sum_{j=i+1}^{k-1} N_j(e_1+e_2)\right], \qquad 1 \le i < k-1$$

Thus,

$$E_i = \begin{cases} N_{k-1}e_1B, & i = k-1 \\ B\left[N_i e_1 + \sum\limits_{j=i+1}^{k-1} N_j(e_1+e_2)\right], & 0 \le i < k-1 \end{cases} \tag{17.3}$$

Ideally, the energy depletion across the network is balanced and the energy efficiency is optimized when all the sensors of the network exhaust their energy at the same time.

Theorem 5 *(Wu et al. [55])* Balanced energy depletion is not achievable in the model under consideration.

It is easy to understand that this impossibility is rooted in the traffic pattern that the sensors in the outermost corona C_{k-1} only need to transmit their own data, but the sensors in the corona C_{k-2} or other inner coronas need to forward their own data as well as those from outer coronas.

17.5.3 Sub-balanced Energy Depletion

We show that it is possible to achieve the sub-balanced energy depletion although it is impossible to attain balanced energy depletion in the network. In fact,

Theorem 6 *(Wu et al. [55])* If the network achieves the sub-balanced energy depletion, then the number of sensors in coronas grows in geometric progression from the outer coronas to the inner ones, except C_{k-1}.

Therefore, the sub-balanced energy depletion of the whole network is possible only if the number of sensors in coronas grows in geometric progression from C_{k-2} to C_0.

Theorem 7 *(Wu et al. [55])* If the number of sensors in coronas increases from C_{k-2} to C_0 in geometric progression with common ratio $q > 1$, and there are $N_{k-2}/(q-1)$ sensors in C_{k-1}, then the network can achieve the sub-balanced energy depletion.

In this section, armed with the above results, we present a novel non-uniform sensor distribution strategy for achieving sub-balanced energy depletion and validate it

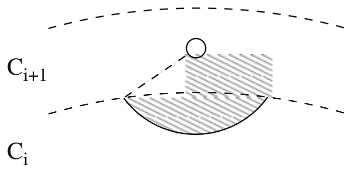


Fig. 17.13 Reachable area in C_i of a sensor in C_{i+1}

by a distributed shortest path routing algorithm, *q-Switch Routing*. Moreover, in order to make easier the discussion of such routing algorithm, we assume that the sensors are always awake. This is achievable in our model by replacing each routing node with a group of adjacent sensors with different types, in such a way that when the sensor in charge of routing goes back to sleep, another awake sensor takes over the routing duty.

The proposed non-uniform sensor distribution strategy regulates the number of sensors in different coronas with the aim of achieving the sub-balanced energy depletion in the network. All the sensors are assumed to have been deployed a priori from the outermost corona to the innermost one such that the number of sensors in the coronas satisfies the following constraint:

$$\frac{N_i}{N_{i+1}} = \begin{cases} q, & q > 1, \quad 0 \le i < k-2 \\ q-1, & i = k-2 \end{cases} \tag{17.4}$$

It is possible that the sensors are assigned in such a way that each sensor in the corona C_{k-1} can communicate directly with $(q-1)$ different sensors in C_{k-2}, and sensors in C_{i+1} can communicate directly with q different sensors in C_i, where $0 \le i < k-2$. In Fig. 17.13, we show that we can deploy corresponding $(q-1)$ or q sensors in the reachable area[3] (the shadow area in the figure) in the next inner corona for each sensor in C_i, unless it is on the border. The processes can be repeated until the deployment in C_0 is finished. Therefore, shortest paths that start from sensors in C_i and end at the sink via i hops can be constructed. We do not assign sensors on the border of any corona, due to the fact that when a sensor is placed there, the reachable area in the adjacent coronas reduces to a point.

Let S_i denote the area of corona C_i. Then the sensors density is given by

$$\delta_i = \frac{N_i}{S_i} = \frac{N_i}{\pi(2i-1)} \tag{17.5}$$

Now, the ratio between the sensor densities of two adjacent coronas C_{i+1} and C_i is

[3] Obviously, q is limited to the maximum number of sensors that can be deployed in the reachable area. On the one hand, we will see later that the network can achieve very high energy efficiency even with a small q, e.g., $q = 2$. On the other hand, the size of a sensor could be insignificant compared with a real field for deployment. Therefore this restriction is not a concern.

$$\frac{\delta_{i+1}}{\delta_i} = \begin{cases} \dfrac{2i-1}{q(2i+1)}, & q > 1, \quad 0 \le i < k-2 \\ \dfrac{2k-5}{(q-1)(2k-3)}, & q > 1, \quad i = k-2 \end{cases} \tag{17.6}$$

This implies that the ratio is only related to the common ratio (q) of the geometric progression and the index of the corona.

17.5.4 q-Switch Routing and Comparison with Other Node Distribution Strategies

We start sketching the q-switch routing algorithm for WSNs used to evaluate the proposed non-uniform sensor distribution strategy. We call it q-Switch Routing, for any sensor in the network has q or $(q-1)$ relay candidates directing to the sink in the adjacent inner corona. We assume that there is a network initialization process in which sensors find their downstream relay candidates and record them. Then there are N_{k-1} q-ary trees formed when the network finishes the initialization process. In order to evenly distribute energy depletion among the relay sensors, the source sensor always selects one relay sensor with maximum residual energy. This can be done by exchanging messages on the energy status with the relays in the neighborhood. The source sensor can switch to next relay candidate sequentially because only one sensor among the q or $(q-1)$ relay candidates has been selected to forward data last time. After the source sensor chooses the relay sensor, it forwards data of its own and those from the upstream sensor or so-called parent sensor. For sensors without data to forward, they just send their own data to the downstream selected relay sensor or so-called child sensor. The chosen relay sensor will repeat this process until the data arrive at a sensor in corona C_1, then the data will be delivered to the sink. The pseudo-code of the routing algorithm is presented in Fig. 17.14. Figure 17.15 illustrates part of the constructed q-ary tree and the data forwarding process.

We have performed simulations to compare the proposed strategy with two other possible node distribution strategies: (i) *non-uniform random node distribution*: the number of nodes in each corona is regulated as the proposed one, but nodes are

```
  Procedure $q$-Switch Routing;
      On receiving a DATA_FORWARD_MSG from sensor $i$
1         $j = SelectNextRelay(q)$;
2         if $IsParent(i) = TRUE$ then
3             $Send(j,$ DATA_FORWARD_MSG$(data))$;
4         else
5             $DiscardMsg$;

      On receiving no message
6         $j = SelectNextRelay(q)$;
7         $Send(j,$ DATA_FORWARD_MSG$(own-data))$;
```

Fig. 17.14 The q-Switch routing algorithm

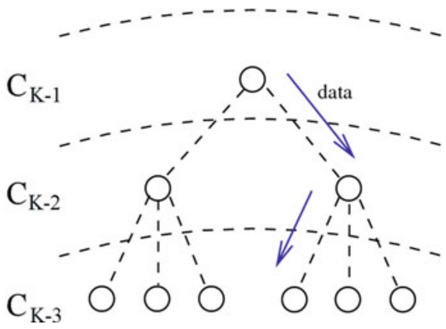


Fig. 17.15 q-Switch Routing, $q = 3$: data traverses from the root to the leaves of the q-ary tree and to the sink eventually

deployed randomly instead of being assigned a priori; and (ii) *uniform node distribution*: nodes can appear at any place with equal probability; different from the two *non-uniform* node distribution strategies, the number of nodes in each corona is not regulated. We deploy eight nodes in the outermost corona when we implement the two *non-uniform* node distribution strategies including the proposed one. The number of nodes in the inner coronas increases in geometric progression with a common ratio of 2 ($q = 2$). We have examined the performance in terms of network lifetime, residual energy ratio, and data delivery ratio with the network radius increasing from 3 to 9.

For the sake of simplicity, we have implemented the routing assuming that the sensors are always awake. We have respectively implemented two similar algorithms as the q-Switch Routing in the networks with non-uniform random node distribution and uniform node distribution for the purpose of comparison. The common basic idea of them is that each node keeps forwarding its data to one of its neighbors with maximum remaining energy.

Figure 17.16 shows the lifetime of the network using the three strategies. We see that networks with uniform node distribution strategy and non-uniform random node distribution strategy decrease with the growth of the network radius, i.e., the number of coronas, while network with the proposed strategy enjoys quite stable lifetime. Note that the strategy of non-uniform random node distribution performs better than that of uniform node distribution, offering some hints on the application of our theory with less difficulty in real node deployment. The fact that the network lifetime maintains longer and steadier shows better scalability of the proposed strategy in terms of network lifetime.

Residual energy ratios of networks using the strategies are shown in Fig. 17.17. We observe that the residual energy ratios of networks with non-uniform random node distribution and uniform node distribution are over three times greater than that of network with the proposed non-uniform node distribution. Nevertheless, network with non-uniform random node distribution performs better than that with uniform node distribution in most simulations. This also implies the effectiveness of the proposed strategy.

While the non-uniform node distribution strategy appears to be quite energy efficient, it incurs some costs. The total number of nodes in the network grows exponentially with the number of nodes increasing from outer coronas to the inner

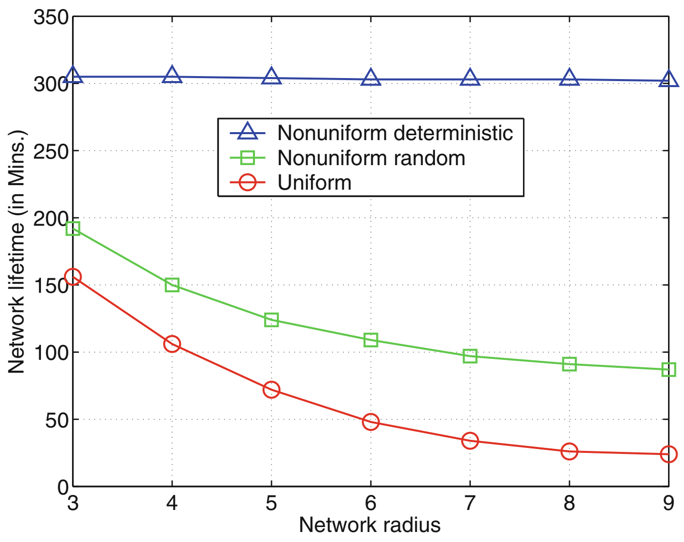


Fig. 17.16 Network lifetime of different node distributions

ones in geometric progression. Consequently, the cost of sensor nodes needs to be reduced significantly in order for this non-uniform node distribution strategy to be more practical.

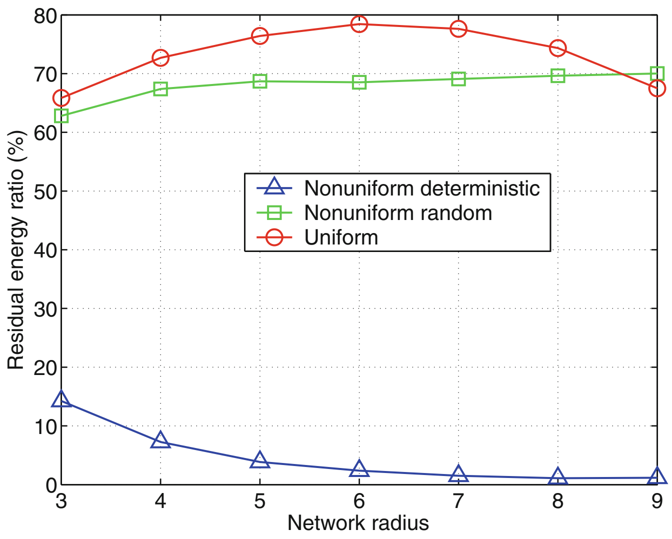


Fig. 17.17 Residual energy ratios of different node distributions

17.6 Concluding Remarks

In this chapter, we have presented two interesting problems in the field of dense and random deployed sensor networks. Namely, we have considered the localization and the energy hole problems in sensor networks deployed in a circular area around a special node, called sink, which is used upstream for localization purposes and downstream for gathering sensed data.

The localization problem deals with the requirement in sensor networks to relate the sensed data with some kind of coordinates of the sensors which accomplished the measurements. In fact, without such a relation, the sensed data might be completely useless. For this purpose, we have presented a series of protocols which provide sensors with coarse-grain coordinate system based on the subdivision of the sink zone (the circular sensor network) into concentric coronas of equal width and equiangular sectors centered at the sink. The Flat– protocol is the simplest one from computational viewpoint because each sensor performs $O(1)$ operations per time slot. In contrast, Flat+ has the better performance, but it cannot be used if sensors are not allowed to skip one or more awake periods. In Flat– as well as in its improvements, we assume an asynchronous model and the sensors never communicate. Concerning the Cooperative protocol, this is one of the few protocols which exploits the high density of sensor networks in favor of a fast and cheap training process. The results presented in this chapter show that the protocols are flexible, in the sense that their parameters can be properly tuned. For instance, fixing the number k of coronas, one can decide the optimal values of d and L so as to minimize the number of sleep/awake transitions and/or the overall awake time per sensor. Conversely, one can fix the desired number of sleep/awake transitions and then select suitable values of d and L.

Concerning the energy hole problem, we have explored the theoretical aspects of the non-uniform node distribution strategy. We find that although it is impossible to achieve balanced energy depletion among all the nodes due to the traffic pattern of WSNs, the sub-balanced energy depletion in the network is possible. We show that with the proposed non-uniform node distribution strategy, the network can achieve very high energy efficiency. We formulate the ratio between the node densities of the adjacent $(i + 1)$th corona and the ith one by this strategy. We present a new routing algorithm called q-Switch Routing which is tailored for the proposed non-uniform node distribution strategy. In conclusion, we have shown how the network density can offer a solution, compatible with our new localization algorithms, for the sink energy hole problem, which is ineluctable in our sensor network model.

Other interesting extensions to the presented model concern different shapes for the deployment area, more sink nodes in the network, and mobility aspects for both sensors and sink. Determining the "best" path for a mobile sink which collects data over all the network might be a decisive factor for prolonging the network lifetime taking into consideration the possibility of varying density. Also considering more collaborative sink nodes deployed or moving in the network might prove vital for the network as well. Finally, providing localization when the sensors move is definitely a challenging problem still open in our model.

References

1. The sensor network museum project: http:// www.snm.ethz.ch/ main/ homepage
2. I. Akyildiz, I. Kasimoglu. Wireless sensor and actor networks: *Research Challenges* 2:351–367, 2004.
3. I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci. Wireless sensor networks: *A survey*. Computer Networks 38(4):393–422, 2002.
4. B. Alavi, K. Pahlavan. Modeling of the toa-based distance measurement error using uwb indoor radio measurements. *IEEE Communications Letters* 10(4):275–277, 2006.
5. J. BachRach, C. Taylor. Localization in sensor networks. In: I. Stojmenovic (ed.) Handbook of sensor networks: *Algorithms and Architectures*. Wiley & Sons, Inc., Hoboken, New Jersey, 2005.
6. F. Barsi, A. Bertossi, F. Betti Sorbelli, R. Ciotti, S. Olariu, M. Pinotti. Asynchronous corona training protocols in wireless sensor and actor networks. *IEEE Transactions on Parallel and Distributed Systems* 20(8): 1216–1230, Los Alamitos, USA, 2009.
7. F. Barsi, A. Bertossi, C. Lavault, A. Navarra, S. Olariu, M. Pinotti, and V. Ravelomanana. Efficient binary search for training heterogeneous sensor and actor networks. In: *Proceedings of the 1st ACM Workshop on Heterogeneous Sensor and Actor Networks* (HeterSANET), pages 17–24, 2008.
8. A. Bertossi, S. Olariu, M. Pinotti. Efficient corona training protocols for sensor networks. *Theoretical Computer Science* 402(1):2–15, 2008.
9. F. Betti Sorbelli, R. Ciotti, A. Navarra, M. Pinotti, and V. Ravelomanana. Cooperative training in wireless sensor and actor networks. In: *Proceedings of the 6th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine)*, pages 569–583, Canary island, Spain, 2009.
10. J. Bruck, J. Gao, A. Jiang. Localization and routing in sensor networks by local angle information. In: *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, ACM Press, pages 181–192, Urbana-Champaign, USA, 2005.
11. N. Bulusu, J. Heidemann, D. Estrin, T. Tran. Self-configuring localization systems: *Design and Experimental Evaluation* 3(1):24–60, 2004.
12. N. Burri, P. von Rickenbach, R. Wattenhofer. Dozer: Ultra-low power data gathering in sensor networks. *In: Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN)*, ACM pages 450–459, Cambridge, Massachusetts, USA, 2007.
13. J. Deng, Y. Han, W. Heinzelman, P. Varshney. Balanced-energy sleep scheduling scheme for high-density cluster-based sensor networks. *Computer Communications* 28(14):1631–1642, 2005.
14. G. Di Stefano, F. Graziosi, F. Santucci. Distributed positioning algorithm for ad-hoc networks. In: *Proceedings of the IEEE International Workshop on UWB Systems*, Virginia, USA, 2003.
15. G. Di Stefano, A. Petricola. A Distributed aOA based localization algorithm for wireless sensor networks. *Journal of Computers* 3(4):1–8, 2008.
16. C. Efthymiou, S. Nikoletseas, J. Rolim. Energy balanced data propagation in wireless sensor networks. *Wireless Networks* 12(6):691–707, 2006.
17. A. Giridhar, P. Kumar. Maximizing the functional lifetime of sensor networks. In: *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 5–12, 2005.
18. T. He, C. Huang, B. Blum, J. Stankovic, T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In: *Proceedings of the 9th International Conference on Mobile Computing and Networking (MobiCom)*, pages 81–95, San Diego, USA, 2003.
19. W. Heinzelman, A. Chandrakasan, H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications* 1(4):660–670, 2002.
20. A. Hopper, A. Jones, A. Ward. A new location technique for the active office. *IEEE Personal Communications* 4(5):42–47, 1997.

21. A. Jarry, P. Leone, O. Powell, J. Rolim. An optimal data propagation algorithm for maximizing the lifespan of sensor networks. In: *Proceedings of the 2nd International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 405–421, San Fancisco, USA, 2006.
22. K. Langendoen, N. Reijers. Embedded systems handbook: Distributed localization algorithms. CRC Press, Boca Raton, Florida, USA, 2004.
23. P. Leone, S. Nikoletseas, J. Rolim. Stochastic models and adaptive algorithms for energy balance in sensor networks. *Theory of Computing Systems* 47(2):433–453, Springer, New York, 2010
24. C. Li, M. Ye, G. Chen, J. Wu. An energy-efficient unequal clustering mechanism for wireless sensor networks. In: *Proceedings of the 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, Washington DC, USA, 2005.
25. J. Li, P. Mohapatra. An analytical model for the energy hole problem in many-to-one sensor networks. In: *Proceedings of the IEEE 62nd Semiannual Vehicular Technology Conference (VTC)*, pages 2721–2725, Dallas, USA, 2005.
26. J. Li, P. Mohapatra. Analytical modeling and mitigation techniques for the energy hole problems in sensor networks. *Pervasive and Mobile Computing* 3(8):233–254, 2007.
27. J. Lian, K. Naik, G. Agnew. Data capacity improvement of wireless sensor networks using non-uniform sensor distribution. *International Journal of Distributed Sensor Networks* 2(2): 121–145, 2006.
28. Y. Liu, H. Ngan, L. Ni. Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting. In: *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, pages 128–135, Taichung, Taiwan, 2006.
29. Z. Lotker, A. Navarra. Grid emulation for managing random sensor networks. *Ad Hoc Networks* 6(6):900–908, 2008.
30. J. Luo, J. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In: *Proceedings of the 24th Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1735–1746, Miami, USA, 2005.
31. G. Mao, B. Fidan. (eds.): Localization algorithms and strategies for wireless sensor networks. IGI Global, Hershey, Pennsylvania, USA, 2009.
32. R. Mudumbai, D. Brown III, U. Madhow, H. Poor. Distributed transmit beamforming: Challenges and recent progressdistributed transmit beamforming: Challenges and recent progress. *IEEE Communications Magazine*, 2:102–110, 2009.
33. A. Navarra, M. Pinotti, V. Ravelomanana, F. Betti Sorbelli, R. Ciotti. Cooperative training for high density sensor and actor networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, Special Issue on Mission Critical Networking Vol. 28(5), pages 753–763, Piscataway, New Jersey, USA, 2010.
34. A. Navarra, A. Tofani. Distributed localization strategies for sensor networks. In: *Proceedings of the 4th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 1–3, 2007.
35. D. Niculescu. Positioning in ad-hoc sensor networks 18(4):24–29, 2004.
36. D. Niculescu, B. Nath. Ad hoc positioning system (APS) using AoA. In: *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, IEEE Computer Society pages 1734–1743, 2003.
37. S. Olariu, I. Stojmenovic. Data-centric protocols for wireless sensor networks. In: I. Stojmenovic (ed.): Handbook of Sensor Networks: Algorithms and Architectures, John Wiley & Sons Inc., Hoboken, New Jersey, USA, 2005.
38. S. Olariu, I. Stojmenovic. Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting. In: *Proceedings of the 25th Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1–12, Barcelona, Catalunya, Spain, 2006.

39. S. Olariu, A. Waada, L. Wilson, M. Eltoweissy. Wireless sensor networks leveraging the virtual infrastructure. *Network* 18(4):51–56, 2004.
40. A. Papadopoulos, A. Navarra, J. McCann. VIBE: A Virtual-infrastructure-based energy-efficient framework for routing over scalable wireless sensor networks. In: *Proceedings of the 1st international Workshop on Energy in Wireless Sensor Networks (WEWSN)*, Santirini island, Greece, 2008.
41. M. Perillo, Z. Cheng, W. Heinzelman. On the problem of unbalanced load distribution in wireless sensor networks. In: *Proceedings of IEEE Global Communications Conference (GLOBE-COM)*, pages 74–79, Texas, USA, 2004.
42. O. Powell, P. Leone, J. Rolim. Energy optimal data propagation in wireless sensor networks. journal of parallel and distributed computing 67(3): 302–317, 2007.
43. J. Rabaey, J. Ammer, T. Karalar, S. Li, B. Otis, M. Sheets, T. Tuan. Pico-radios for wireless sensor networks: The next challenge in ultra-low power design. In: *Proceedings of the IEEE International Solid-State Circuits Conference - Digest of Technical Papers (ISSCC)*, pages 156–445, Pennsylvania, USA, 2002.
44. T. Rappaport. Wireless communications: Principles and practice. Prentice-Hall, New Jersey, USA, 1996.
45. M. Rudafshani, S. Datta. Localization in wireless sensor networks. In: *Proceedings of the 6th Localization in Wireless Sensor Networks (IPSN)*, pages 51–60, 2007.
46. A. Savvides, L. Girod, D. Estrin. Localization in sensor networks. Wireless sensor networks pages 327–349, 2004.
47. G. Shi, M. Liao, M. Ma, Y. Shu. Exploiting sink movement for energy-efficient load-balancing in wireless sensor networks. In: *Proceeding of the 1st ACM international workshop on Foundations of wireless ad hoc and sensor networking and computing* (*FOWANC*), pages 39–44, New Orleans, USA, 2008.
48. H. Shiue, G. Yu, J. Sheu. Energy hole healing protocol for surveillance sensor networks. In: *Workshop on Wireless, Ad Hoc, and Sensor Networks* (2005)
49. C. Song, J. Cao, M. Liu, Y. Zheng, H. Gong, G. Chen. Mitigating energy holes based on transmission range adjustment in wireless sensor networks. In: *Proceedings of the 5th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine)*, pages 1–7, Hong Kong, China, 2008.
50. S. Soro, W. Heinzelman. Prolonging the lifetime of wireless sensor networks via unequal clustering. In: *Proceedings of the 19th International Parallel and Distributed Processing Symposium (IPDPS)*, Denver, Colorado, USA, 2005.
51. N. Szabo, R. Tanaka. Residue arithmetic and its applications to computer technology. McGraw-Hill, New York, USA, 1967.
52. A. Waada, S. Olariu, L. Wilson, M. Eltoweissy, K. Jones. Training a wireless sensor network. *Mobile Networks and Applications* 10(1):151–168, 2005.
53. D. Wang, B. Xie, D. Agrawal. Coverage and lifetime optimization of wireless sensor networks with gaussian distribution. *IEEE Transactions on Mobile Computing* 7(12):1444–1458, 2008.
54. W. Wang, V. Srinivasan, K. Chua. Using mobile relays to prolong the lifetime of wireless sensor networks. In: *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking* (*MobiCom*), pages 270–283, Cologne, Germany, 2005.
55. X. Wu, G. Chen, S. Das. Avoiding energy holes in wireless sensor networks with nonuniform node distribution. *IEEE Transactions on Parallel and Distributed Systems* 19(5): 710–720, 2008.
56. Q. Xu, R. Ishak, S. Olariu, S. Salleh. On asynchronous training in sensor networks. In: *Proceedings of the 3rd International Conference on Advances in Mobile Multimedia (MoMM)*, pages 43–50, 2005.
57. M. Ye, C. Li, G. Chen, J. Wu. EECS: an energy efficient clustering scheme in wireless sensor networks. *International Journal of Ad Hoc and Sensor Wireless Networks* 3(2–3):99–119, 2007.
58. J. Yick, B. Mukherjee, D. Ghosal. Wireless sensor network survey. *Computer Networks* 52(12):2292–?2330, 2008.

59. O. Younis, S. Fahmy. HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on Wireless Communications* 3(4):660–669, 2004.
60. Z. Zeng, A. Liu, Z. Chen, X. Wu, J. Long. Improved analysis of energy hole for wireless sensor networks. In: *Proceedings of the 1st International Conference on Communications and Mobile Computing* (*CMC*), pages 553–557, 2009.
61. H. Zhang, H. Shen. Balancing energy consumption to maximize network lifetime in data-gathering sensor networks. *IEEE Transactions on Parallel and Distributed Systems* 20(10):1526–1539, 2009.
62. R. Zhang, Z. Jia, D. Yuan. Analysis of lifetime of large wireless sensor networks based on multiple battery levels. *International Journal of Communications, Network and System Sciences* 1(2):136–143, 2008.

Chapter 18
Prolong the Lifetime of Wireless Sensor Networks Through Mobility: A General Optimization Framework

Jun Luo and Liu Xiang

Abstract Though mobility is rarely considered in traditional *wireless sensor networks* (WSNs), actively exploiting mobility to improve the performance of WSNs has been increasingly recognized as an important aspect of designing WSNs. This chapter focuses on exploiting mobility to improve the network lifetime of a WSN. We present a general optimization framework that is able to capture several aspects of maximizing network lifetime (MNL) involving mobile entities. Based on this framework, we conduct an in-depth analysis on each of these aspects and also describe algorithms that can be used to solve the resulting optimization problems. We also present certain numerical results where engineering insights can be acquired.

18.1 Mobile Elements in Wireless Sensor Networks: Stir Up the Pond

Traditionally, mainstream research envisioned *wireless sensor networks* (WSNs) as an avatar of **static** multi-hop wireless networks [1]. Although the mobility issues were present even from the early stage of the WSN-related investigations (e.g., [27]), those issues failed to attract a lot of attention until very recently. The reason is twofold. On one hand, it is much more difficult, from both theoretical and practical point of views, to deal with networks with mobile entities. On the other hand, we only recently realized that we could **actively** utilize mobility rather than having to **passively** accept its inevitable presence. In this chapter, we will focus on one of the active applications of mobility to improve an important aspect of network performance — *lifetime*. However, instead of directly addressing the main topic, we will start with a brief survey of various aspects of WSNs that are concerned with mobility, which should provide the readers with a better technological context in understanding the main topic.

J. Luo (✉)
School of Computer Engineering, Nanyang Technological University (NTU), Singapore
e-mail: junluo@ntu.edu.sg

Typically, mobility gets involved in WSNs in two ways: either passively or actively. In the former case, mobility comes as input to certain system design aspects of WSNs, and a certain design has to cope with the negative effects (e.g., unreliable communication channels and high cost of route maintenance) brought by mobility. Typical instances of this case exist where either *sensor nodes*[1] or *sinks*[2] need to move according to the application requirements: for example, a sensor node or a sink may be attached to a tactic unit in a battle field [16, 37], or a sink is someone's PDA that helps him/her to navigate within a sensor field [20]. Another approach (e.g., [17]) exploits the anyway present mobility as an efficient replacement for connectivity and data propagation redundancy. In the latter case, mobility is actively introduced to a network by system designers, aiming at improving certain performance aspects of the original design that consists of only static network components. Here, both theoreticians and practitioners are trying to make the best use of mobility while still coping with its side-effects. Typical performance aspects that may benefit from the introduction of mobile entities are load balancing/lifetime maximization [7, 21, 22, 24, 30, 32, 35, 36], buffer overflow prevention [15, 33], coverage enhancement [34], and high fidelity data collection [3].

Although many performance aspects of WSNs may benefit from mobility, the lifetime issue seems to have attracted the majority of attention and contributions. Therefore, we focus on the issue of prolonging network lifetime using mobility in this chapter. As shown in Fig. 18.1, the traffic load within a WSN is highly unbalanced among nodes that have different distances from the sink. Whereas no routing strategy may alleviate such an imbalance, actively moving certain network entities may further balance the load and hence improve the lifetime. Basically, two approaches, namely *fast mobility* and *slow mobility*, are used to exploit entity (sink or node) mobility to improve network lifetime. They are distinguished by the relationship between the moving speed of an entity and the tolerable delay of the data delivery. In the former case, an entity (typically a sink) can transport data with its mechanical movements if its speed is sufficiently high so that the mechanical data transportation yields a tolerable data delivery delay. In this case, nodes may be totally or partially spared from the traffic forwarding load and can hence save their energy. We term this approach **fast** mobility approach, as the entity should move at a sufficiently high speed. In the latter case, moving an entity, even very infrequently (say once a day or week), may still benefit the network lifetime, thanks to the distribution of the role of bottleneck nodes within the entire network. We denote this approach as **slow** mobility, because the moving speed of a mobile entity is too low to be used for transporting data within a tolerable delay (but it barely affects the delay due to the way it is used).

The general reason that mobility, no matter fast or slow, can improve network lifetime lies in the fact that mobility increases the dimension (thus the degree of freedom) of the problem. This follows the general principle that optimizing an

[1] In this chapter, the words *sensor node* and *node* are used interchangeably.
[2] These are the entities that collect data from WSNs; sometimes they are also termed *base stations*.

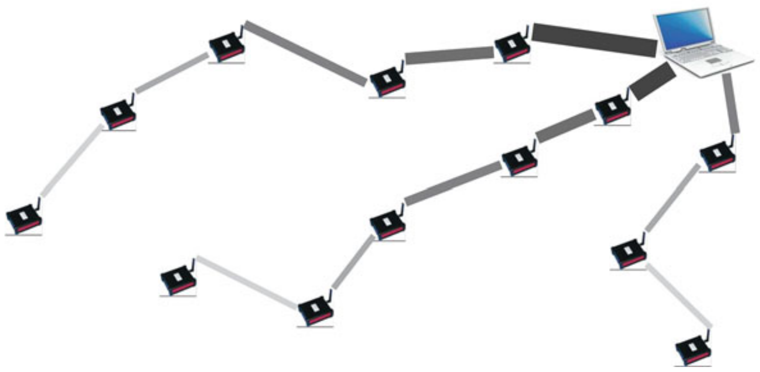


Fig. 18.1 The unbalanced traffic load in a WSN due to the converging traffic pattern that accumulates traffic toward the "last hop" nodes

objective in a high-dimension space always leads to a result no worse than what can be achieved in a subspace of reduced dimension. However, solving problems in high-dimension space incurs a higher complexity. In the remainder of this chapter, we will discuss a general optimization framework that can be used to model and formulate such problems, and we will highlight the solution techniques that are used.

Under the slow mobility regime, the mobility may take a discrete form: the movement trace consists of several anchor points between which the mobile entities move and at which they pause. Consequently, data packets have to be carried from their origins to the sinks through multi-hop routing. In Sects. 18.2 and 18.3, we discuss the approach that makes use of the slow sink mobility to balance the traffic load within a WSN and hence to improve the network lifetime, and we introduce a general optimization framework to model, formulate, and solve the problem. The approach considered in Sect. 18.2 aims at obtaining or approximating the optimal movement traces of multiple mobile sinks, but the anchor points that constitute the traces can only be chosen from a predefined set of locations. The extension reported in Sect. 18.3 takes one step further by relaxing the location constraint: should the algorithm in Sect. 18.2 obtain an optimal solution, an optimal unconstrained trace could be obtained exactly or be approximated to an arbitrarily small granularity, at a cost of solving many instances of the problem addressed in Sect. 18.2.

In Sect. 18.4, we present an approach that extends the general framework discussed in Sect. 18.2 to a distinct direction. This approach, though still categorized as a slow mobility approach, chooses to move certain powerful (in terms of energy reserve) nodes rather than sinks. The underlying rationale is to use these powerful (mobile) nodes to replace certain highly loaded (static) nodes from time to time, which could substantially reduce the energy consumption of those static nodes and hence prolong the network lifetime. Although it has been shown that this *mobile node* approach is in general inferior to the *mobile sink* approaches discussed in

Sects. 18.2 and 18.3, the proposal is still meaningful because moving sink(s) might not always be feasible. Also, as shown by our numerical results in Sect. 18.4.3, the performance of this approach can be comparable to that of the mobile sink approach.

Under the fast mobility regime, entities may move fast enough to deliver data with a tolerable delay, WSNs can hence take advantage of mobility capacity [14]. We term this approach *mobile relay*, although the mobile entities are the sinks, because the mobile sink, instead of only receiving multi-hop transmissions, may "pick up" data from nodes (through one-hop transmissions) and transport the data with mechanical movements. In Sect. 18.5, we first extend our framework to show the complexity of finding an optimal tour for the mobile relay, and then we will introduce a possible simplification of the problem along with the approximation algorithms designed for a single mobile relay.

Although the algorithms we discuss are centralized, they serve as benchmarks or guidance for distributed implementations (e.g., [7, 24]). The intention of this chapter is to give an in-depth treatment on the issue of using mobility to prolong network lifetime from the theoretical perspective, so we do not claim any thoroughness in surveying the vast literature on the mobility-related issues in WSNs; such literature is too vast even for this specific topic.

18.2 Balancing Traffic Load with Mobile Sinks: The Case of Constrained Mobility

As shown in Fig. 18.1, it is the converging traffic pattern of WSNs that leads to the unbalanced traffic load within a network. Consequently, simply manipulating the routing protocols would not fully address the lifetime issue. Fortunately, recent proposals (e.g., [11, 21]) suggest that moving the sinks could distribute the role of bottleneck nodes (those close the sinks) over time and thus even out the load, as illustrated in Fig. 18.2. To support the feasibility of such an approach, a simple implementation is also reported later [23]. In this section, we are aiming at developing an optimization framework to analyze the problem and also a solution technique to solve the optimization problem. In addition, we are making this framework sufficiently general such that we can extend it in different directions later on.

18.2.1 Network Model and Problem Formulation

For a WSN, we use set $\mathcal{N} : |\mathcal{N}| = n$ to represent all the sensor nodes and set $\mathcal{S} : |\mathcal{S}| = m<n$ for the sinks. The former set is static and it determines the basic topology of the network, while the latter changes its layout occasionally so as to collect data and to balance the traffic load. We allow the sinks to choose their locations only within a finite set $\mathcal{V}$. We denote by *on-graph* mobility the case $\mathcal{V} = \mathcal{N}$, and by *off-graph* mobility the case $\mathcal{V} \supset \mathcal{N}$. There is a cost assignment $\mathbf{c} : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}^+$,

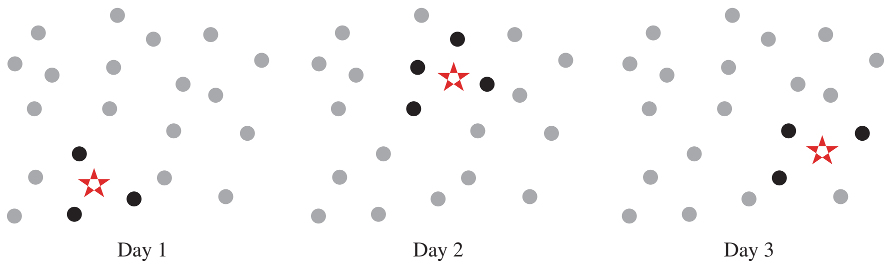


Fig. 18.2 Using a mobile sink to balance the traffic load within a WSN. The star represents the sink, and nodes with a darker color are the bottleneck nodes. Note that the mobility is slow, as the sink may change it location very infrequently

such that a link (i, j) exists (or $\exists(i, j) \in \mathcal{E}$) if and only if (1) $i \in \mathcal{N}$, (2) $j \in \mathcal{N}$ or j is a sink, and (3) the transmission energy[3] e_i^{T} of node i is no less than $c(i, j)$. All these allow us to model the WSN as a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We assume that wireless communication is the dominating energy-consuming factor and hence omit other energy consuming functions such as sensing. We emphasize the crucial behaviors of mobile sinks in our investigation: each sink travels among a set of locations chosen within $\mathcal{V}$ and stays with each one of them to collect data from the whole WSN for a relatively long time, which makes the traveling time negligible. When co-located with a node $i \in \mathcal{N}$, the sink, apart from collecting data from other nodes, inherits all the energy-consuming functions of the co-located node.

Initialized with an energy reserve E_i for each node $i \in \mathcal{N}$, the network is said to be "dead" once some node runs out of battery. In other words, the network lifetime T is defined as the time when the first node dies [6]. Taking into account the fact that sinks change their locations from time to time, we define an *epoch* as a time duration within which no sink changes its position. Thus T can be represented by the summation of time duration of each epoch t_k. In formulating the *maximizing network lifetime* (MNL) problem, we consider the constraints related to $\mathcal{V}$ and $\mathcal{S}$ separately: each node $i \in \mathcal{V}$ is associated with a flow conservation and an energy conservation, while each sink $s \in \mathcal{S}$ is constrained by its location choices. The two sets of constraints are coupled by an indicator matrix $\left[\delta_{is}^k\right]$ where $\delta_{is}^k = 1$ if sink s is co-located with node $i \in \mathcal{V}$ during the kth epoch and $\delta_{is}^k = 0$ otherwise. We formally present the mixed-integer nonlinear programming of MNL as below, and the detailed notations can be found in Table 18.1.

[3] The physical features of the radio of node i are usually specified by a tuple (P_i, R, μ); here P_i is the transmission power, R is the data rate, μ is the threshold (specified by the required *bit error rate* (BER) of a given modulation scheme that produces the rate R) such that a link (i, j) may operate on rate R iff $P_i \cdot \eta_{i,j} \geq \mu$, where $\eta_{i,j}$ represents the fading, shadowing, and path loss effects between nodes i and j. Our model can be considered as a more generalized form of the aforementioned model, as $e_i^{\mathrm{T}} = \dfrac{P_i}{R} \geq \dfrac{\mu}{R\eta_{i,j}} = c(i, j)$ is indeed the criterion to indicate the existence of link (i, j). Note that, under our model, e_i^{T} may have a unit of, for example, Joules/Bit.

Table 18.1 Notations used through out this section

Notations of The Network	
$\mathcal{N}$	The set of sensor nodes in the WSN
n	$= \lvert\mathcal{N}\rvert$, the number of sensor nodes in the WSN
$\mathcal{S}$	The set of sinks in the WSN
m	$= \lvert\mathcal{S}\rvert$, the number of sinks in the WSN
$\mathcal{V}$	$\supset \mathcal{N}$, the potential locations of the mobile sinks
$\mathcal{E}$	The set of all feasible wireless links
$\mathbf{c}$	Cost assignment that defines feasible links
T	Network lifetime
$\hat{T}$	Maximum network lifetime
t_k	The duration of the kth epoch
Notations of Sensor Nodes	
E_i	Initial energy reserve of node i
e_i^{T}	Energy consumption for node i to transmit a unit of data
e^{R}	Energy consumption for any node to receive a unit of data
λ_i	Information generation rate of node i
r_{ij}^k	Data rate from node i to node j during the kth epoch
r_i^k	Data rate drained out of the WSN from node i during the kth epoch
q_{ij}^k	Quantity of data from node i to node j during the kth epoch
q_{is}^k	Quantity of data from node i to sink s during the kth epoch
P_{is}^k	The set of paths from node i to sink s during the kth epoch
P_i^k	The set of paths going through node i during the kth epoch
Notations of Sinks	
$\mathcal{L}_k$	$\subset \mathcal{V}$, the set of sink locations during the kth epoch
δ_{is}^k	Indicator for the location of sink s during the kth epoch
sl_k	$= \left[\delta_{is}^k\right]_{i\in\mathcal{V},s\in\mathcal{S}}$: the sink layout during the kth epoch

$$\text{maximize} \quad T = \sum_k t_k \tag{18.1}$$

$$\begin{aligned}
\text{subject to} \quad \sum_{(i,j),(j,i)\in\mathcal{E}} \left(r_{ij}^k - r_{ji}^k\right) + r_i^k \delta_{is}^k &\geq \lambda_i && \forall i,k && (18.2)\\
\sum_{k:\delta_{is}^k\neq 1}\left[\sum_{(i,j),(j,i)\in\mathcal{E}} \left(r_{ij}^k e_i^{\mathrm{T}} + r_{ji}^k e^{\mathrm{R}}\right)\right] t_k &\leq E_i && \forall i && (18.3)\\
\sum_{s\in\mathcal{S}} \delta_{is}^k &\leq 1 && \forall i,k && (18.4)\\
\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{S}} \delta_{is}^k &= m && \forall k && (18.5)\\
t_k, r_{ij}^k, r_i^k &\geq 0 && \forall i,j,s,k && (18.6)\\
\delta_{is}^k &\in \{0,1\} && \forall i,s,k && (18.7)
\end{aligned}$$

By this formulation, we implicitly assume that the data rate between any two nodes i and j, r_{ij}^k, is feasible under the corresponding link capacity; otherwise, we can always make it feasible by adjusting the data generation rate vector $\Lambda = [\lambda_i]$. Note that the data can be drained out from node or location i only if a sink s happens to be there, i.e., $\delta_{is}^k = 1$; otherwise that draining rate equals zero. We also assume that all nodes use an identical receiving power e^{R}, whereas the transmitting power e_i^{T} is set by a node i according to, for example, certain topology control mechanisms [18, 19]. Therefore, transmission and reception together contribute to the energy consumption of a node, and the energy consumed by other activities (e.g., data sensing) is considered as negligible. Finally, as a sink inherits the functions of a co-located node, we do not count the energy consumption of that node during the epoch when there is a sink co-located with it. We denote this phenomenon *substitution effect*, and we will discuss it in detail later on.

In the remainder of this section, we will first analyze the complexity of MNL and derive a duality theory to characterize the optimal solution of the problem. We will then use a simplified version of MNL to motivate a polynomial-time algorithm. Finally, we show that the polynomial-time algorithm can be used to approximate MNL with a provable ratio.

18.2.2 Complexity Analysis of MNL

Merging the explicit sink location constraints (18.4) and (18.5) into the conservation constraints for sensor nodes, we can reformulate MNL into the Arc-Flow form:

$$\text{maximize} \quad T = \sum_k t_k \tag{18.8}$$

$$\text{subject to} \quad \sum_{(i,j),(j,i)\in\mathcal{E}} \left(q_{ij}^k - q_{ji}^k\right) \geq \lambda_i t_k \quad \forall k, i \notin \mathcal{L}_k \tag{18.9}$$

$$\sum_{k:i\notin\mathcal{L}_k} \left[\sum_{(i,j),(j,i)\in\mathcal{E}} \left(q_{ij}^k e_i^{\mathrm{T}} + q_{ji}^k e^{\mathrm{R}}\right) \right] \leq E_i \quad \forall i \tag{18.10}$$

$$t_k, q_{ij}^k \geq 0 \quad \forall i, j, k \tag{18.11}$$

where $q_{ij}^k = r_{ij}^k t_k$ represents the amount of data going from node i to node j during the kth epoch, and $\mathcal{L}_k \subset \mathcal{V}$ indicates the set of sink locations during that period.

This seemingly simple formulation hides the actual complexity of MNL: There could be a tremendous number of possible $\mathcal{L}_k$, because, to place m sinks on $|\mathcal{V}|$ possible positions, we have $\binom{|\mathcal{V}|}{m}$ choices of $\mathcal{L}_k$, which means the numbers of variables and constraints of MNL problem are both exponential in n. This motivates us to formally evaluate the complexity of MNL in the following. Let us reformulate the MNL problem into a Path-Flow form:

$$\text{maximize} \quad T = \sum_k t_k \tag{18.12}$$

$$\text{subject to} \qquad \sum_s \sum_{p \in P_{is}^k} f(p) \geq \lambda_i t_k \quad \forall k, i \notin \mathcal{L}_k \tag{18.13}$$

$$\sum_{k: i \notin \mathcal{L}_k} \left[\sum_{p \in P_i^k} f(p) \left(e_i^{\mathrm{T}} + \mathrm{I}_{p \notin P_{is}^k, \forall s} \cdot e^{\mathrm{R}} \right) \right] \leq E_i \quad \forall i \tag{18.14}$$

$$t_k, f(p) \geq 0 \quad \forall k, p \tag{18.15}$$

where I_A is the indicator function of event A, p is a path between a node and a sink, and $f(p)$ is the flow going through that path. Furthermore, we denote by P_{is}^k the path set from node i to sink s and by P_i^k the set of paths going through node i, both in the kth epoch. Note that the data originated at a node may split into several fractions and flow to different sinks via various paths simultaneously, according to the multi-path formulation of the routing strategy. Since the primal formulation of MNL (18.13) and (18.14) is a linear program, the strong duality holds and hence we could instead investigate the dual problem of MNL shown as follows.

$$\text{minimize} \quad G(\mathbf{w}) = \sum_i E_i w(i) \tag{18.16}$$

$$\text{subject to} \qquad \sum_i \lambda_i W(i,k) \geq 1 \quad \forall k \tag{18.17}$$

$$\sum_{j \in p \in P_{is}^k, j \notin \mathcal{L}_k} w(j) \left(e_j^{\mathrm{T}} + \mathrm{I}_{j \neq i} \cdot e^{\mathrm{R}} \right) - W(i,k) \geq 0 \quad \forall i, k, s, p \tag{18.18}$$

$$w(i), W(i,k) \geq 0 \quad \forall i, k \tag{18.19}$$

where the $w(i)$ is the weight assigned to node i, representing the marginal cost of using an additional unit energy of node i; and $W(i,k)$ is the weight of a commodity, i.e., data flow going from node i to all possible destination sinks during epoch k; it indicates the marginal cost of rejecting a unit demand of the commodity. We can interpret the dual problem as follows: given the optimal solution $\hat{T}$ of the primal, if there existed a solution T' longer than $\hat{T}$, the benefit from the prolonged time period $T' - \hat{T}$ would not cover the cost of maintaining the network for that time period, as either performing the data routing (18.18) or not (18.17) would at least offset the benefit. As a result, the dual formulation implies that such T' should not exist.

As the dual objective is to minimize $G(\mathbf{w})$, implying that $w(i)$ is preferred to be as small as possible. However, we cannot make it as small as we want since it is bounded in (18.18) by $W(i,k)$, which is in turn constrained in (18.17). Thus we conduct the variable elimination by plugging (18.18) into (18.17). For an arbitrary vector $\mathbf{w} = [w(i)]$, the overall cost of rejecting a demand of the commodity from node i to any sink s, according to (18.18), can at most be the minimum transmission cost from i to the destination sinks. Therefore, we set

$$W(i,k) = \sum_{j \in \min\left\{p \mid p \in P_{is}^{k}, s \in \mathcal{S}\right\}, j \notin \mathcal{L}_k} w(j)\left(e_j^{\mathrm{T}} + \mathrm{I}_{j \neq i} \cdot e^{\mathrm{R}}\right) \tag{18.20}$$

Using (18.20) to eliminate $W(i,k)$ in (18.18), we get the combined new constraint as

$$\sum_i \lambda_i \left(\sum_{j \in \min\left\{p \mid p \in P_{is}^{k}, s \in \mathcal{S}\right\}, j \notin \mathcal{L}_k} w(j)\left(e_j^{\mathrm{T}} + \mathrm{I}_{j \neq i} \cdot e^{\mathrm{R}}\right) \right) \geq 1 \quad \forall k \tag{18.21}$$

Actually, an arbitrary vector $\mathbf{w}$ may violate (18.21) and in turn (18.17) and hence be infeasible. But if we can find the most violated constraint and scale up $\mathbf{w}$ accordingly, we are always able to turn it into a feasible solution. More specifically, suppose there is an oracle $\rho(\mathbf{w})$, which is the minimum value of the LHS of (18.21) over k,

$$\rho(\mathbf{w}) = \min_k \left[\sum_i \lambda_i \left(\sum_{j \in \min\left\{p \mid p \in P_{is}^{k}, s \in \mathcal{S}\right\}, j \notin \mathcal{L}_k} w(j)\left(e_j^{\mathrm{T}} + \mathrm{I}_{j \neq i} \cdot e^{\mathrm{R}}\right) \right) \right] \tag{18.22}$$

then testing $\rho(\mathbf{w})<1$ or not will suggest the feasibility of $\mathbf{w}$. This is called *separation oracle* in the terminology of linear programming [26]. If $\rho(\mathbf{w})<1$, we can scale up $\mathbf{w}$ and $W(i,k)$ by $\rho^{-1}(\mathbf{w})$ to make them feasible under the constraints. As a result, we transform the dual problem of MNL into an equivalent one, which is to find a vector $\mathbf{w}$ to minimize $\dfrac{G(\mathbf{w})}{\rho(\mathbf{w})}$. Unfortunately, the oracle is not easy to compute; we hereby show it is actually an NP-complete problem for on-graph mobility ($\mathcal{N} = \mathcal{V}$), which implies that it is NP-hard for off-graph mobility ($\mathcal{N} \subset \mathcal{V}$). Let $K = |\mathcal{S}| = m$, $\omega(i) = \lambda_i$, $\ell(j) = w(j)\left(e_j^{\mathrm{T}} + \mathrm{I}_{j \neq i} \cdot e^{\mathrm{R}}\right)$, and $d(i) = \sum_{j \in \min\left\{p \mid p \in P_{is}^{k}, s \in \mathcal{S}\right\}, j \notin \mathcal{L}_k} \ell(j)$, then the separation oracle is equivalent to the following decision problem:

INSTANCE: A graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, a weight assignment $\omega(i) : \mathcal{N} \to \mathbb{R}_0^+$, a length assignment $\ell(i) : \mathcal{N} \to \mathbb{R}_0^+$, positive integer $K \leq |\mathcal{N}|$, and positive real number B.

QUESTION: Is there a set $\mathcal{P}$ of K points on $\mathcal{G}$ such that, if $d(i)$ is the length of the shortest path from i to the closest point in $\mathcal{P}$, then $\sum_i \omega(i) \cdot d(i) \leq B$?

This is known as the p-median problem and is NP-complete [12]. As stated in [26] (Theorem 3), a linear programming problem is NP-hard if the corresponding separation oracle problem is NP-complete, hence we conclude that:

Proposition 1 *The MNL problem is NP-hard.*

18.2.3 Duality Theory and TMNTM

Before developing the algorithm to solve MNL, we will first harvest the benefit coming with the primal–dual interpretation provided in Sect. 18.2.2: it helps us to build the related duality theory, and it also allows us to address the TMNTM decision problem stated as follows:

> TO MOVE OR NOT TO MOVE (TMNTM): Is there a sink layout schedule $\{(sl_k, t_k)\}$ (sl_k is a vector of $\left[\delta^k_{is}\right]$) such that the lifetime $T = \sum_k t_k$ is longer than what is achieved by any fixed layout *sl*?

This was never fully addressed in the previous work such as [11, 21].

We recapitulate the observation that we make on the dual problem of MNL in the following theorem:

Theorem 1 (MAX-LIFETIME MIN-POTENTIAL RATIO THEOREM) *Given the lifetime maximization problem formulated in (18.12)–(18.15), the optimal lifetime $\hat{T}$ is such that*

$$\hat{T} = \min_{\mathbf{w}} \left[\frac{G(\mathbf{w})}{\rho(\mathbf{w})} \right]$$

where $G(\mathbf{w}) = \sum_i E_i w(i)$ is a linear combination of the energy reserves of all nodes with coefficients $w(i)$, and

$$\rho(\mathbf{w}) \equiv \min_k \rho_k(\mathbf{w}) = \min_k \left[\sum_i \lambda_i \left(\sum_{j \in \min\left\{p | p \in P^k_{is}, s \in \mathcal{S}\right\}, j \notin \mathcal{L}_k} w(j) \left(e^{\mathrm{T}}_j + \mathrm{I}_{j \neq i} \cdot e^{\mathrm{R}} \right) \right) \right]$$

is the minimum "potential" (computed as the sum of the minimum "cost," given $w(i)$, to route λ_i from node i to one of the m centers) achieved among all possible center layouts (or sink layouts) $\{\mathcal{L}_k\}$.

Likewise, for a fixed scheduling (or static sinks), the analogous theorem (first given in [29] and improved in [13]) is cited below:

Theorem 2 (MAX-FLOW MIN-DISTANCE RATIO THEOREM) *Given the maximizing lifetime problem formulated in (18.12)–(18.15) but with a fixed schedule consisting of only one element (sl, t), the optimal lifetime $\hat{T}_{sl}$ is such that*

$$\hat{T}_{sl} = \min_{\mathbf{w}} \left[\frac{G(\mathbf{w})}{\rho_k(\mathbf{w})} \right]$$

where $G(\mathbf{w})$ and $\rho_k(\mathbf{w})$ are defined in the previous theorem, and the center layout is defined by sl.

One can easily see that **Theorem 1** is a non-trivial extension of **Theorem 2**, which is in turn extended from the MAX-FLOW MIN-CUT theorem of Ford and Fulkerson [10] for a single s-t flow. Equipped with these two theorems, we are now ready to answer TMNTM.

Proposition 2 *For on-graph mobility, $\hat{T} > \hat{T}_{fs}$, where $\hat{T}_{fs} = \max_{sl} \hat{T}_{sl}$. Literally, the answer to the TMNTM decision problem is positive.*

Proof Assume that $\hat{T}_{sl} > 0$ is the optimal solution for a certain sl, and $\mathbf{w}^*_{sl}$ is the corresponding weight assignment. By plugging $\mathbf{w}^*_{sl}$ into the dual problem of MNL (18.16-18.19), we can always identify a violated constraint with the oracle that computes $\min_k \rho_k \left(\mathbf{w}^*_{sl}\right)$. For instance, assume that one of the sinks is co-located with i and its most loaded neighbor is j. We know that (18.14) is active for j; otherwise it contradicts the optimality of $\hat{T}_{sl}$. Applying complementary slackness, we have $\rho_i \left(\mathbf{w}^*_{sl}\right) = 1$ (by $\hat{T}_{sl} > 0$), $w^*_{sl}(i) = 0$ (by the fact that (18.14) is inactive for i due to the substitution effect defined in Sect. 18.2.1), and $w^*_{sl}(j) > 0$ (by the fact that (18.14) is active for j and j is the bottleneck of all the paths passing through it). The potential $\rho_j \left(\mathbf{w}^*_{sl}\right)$ is bound to be less than 1, because, by moving the sink from i to j, we shorten the length of some paths by $w^*_{sl}(j)$ without increasing the length of other paths going through i. Therefore, we identify that $\mathbf{w}^*_{sl}$, as the dual solution, is infeasible. Consequently, according to the principle of *certificate of optimality*, we know that $\hat{T}_{sl}$, as the primal solution for the fixed schedule case, is not optimal for the MNL problem and thus $\hat{T} > \hat{T}_{sl}$. Let $\hat{T}_{fs} = \max_{sl} \hat{T}_{sl}$, we also have $\hat{T} > \hat{T}_{fs}$. Q.E.D.

Note that this proof implicitly assumes that the minimum potential $\min_k \rho_k \left(\mathbf{w}^*_{sl}\right)$ and the maximum lifetime $\hat{T}_{sl}$ for a fixed sink layout is computable. As we show in Sect. 18.2.2, however, the separation oracle problem is NP-complete. Also, results in [4] suggest that computing $\hat{T}_{sl}$ is NP-hard. Therefore, **Proposition 2** only gives a qualitative comparison rather than a quantitative one.

Interestingly, the proof of **Proposition 2** stresses the importance of a hidden factor behind the evident load balancing effect, namely the substitution effect. Recall from the model description in Sect. 18.2.1, we assume that whenever a sink is colocated with a node, it inherits all the functions of that node, i.e., it takes the place of that node in the network and hence saves the energy consumption for that node. While the load balancing effect is the driving force behind a significant lifetime improvement, the substitution effect, as presented in the above proof, makes moving sinks superior to keeping them static if the sinks are constrained to be on-graph. In Sect. 18.4, we will discuss an extension that fully exerts the substitution effect to improve lifetime.

It is also worth noting that **Proposition 2** holds only for on-graph mobility. In Figure 18.3, we give two examples showing that a static sink layout is already

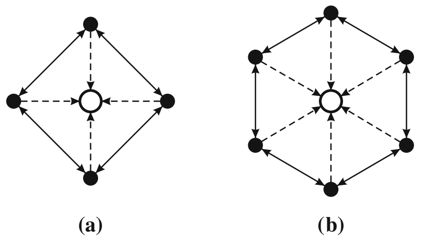


Fig. 18.3 Two examples to show that **Proposition 2** might not hold if sink locations can be off-graph. The *solid lines* represent the original links of the ring networks, and the *dash lines* represent the new links introduced by an off-graph sink (located at the optimal position). It is straightforward to see that sink mobility does not help in these cases: (**a**) 4-node ring and (**b**) 6-node ring

the optimal choice for certain network topologies. Fortunately, we might not have such pathological scenarios in practice. Even if such a case occurs, the optimal (off-graph) sink location might not be available (we refer to [23] for a practical example we have experienced). All the examples we give in Sect. 18.2.5 confirm that moving the sink, no matter on-graph or off-graph, is always superior to keeping it static.

18.2.4 A Primal–Dual Algorithm to Solve MNL

It is straightforward to see that, if there is only one mobile sink, the MNL problem is solvable in polynomial time, because the separation oracle is a P problem. However, directly solving it is practically ineffective on all but very small-scale problems (similar to the case of the concurrent flow problem [29]). In addition, common techniques such as the interior point or simplex algorithms cannot be extended to address the MNL problem involving multiple mobile sinks. In Sect. 18.2.4.1, we will discuss a primal–dual algorithm that solves the MNL problem with a single mobile sink efficiently. Moreover, we will extend the algorithm to approximate the solution of the general MNL in Sect. 18.2.4.2.

18.2.4.1 MNL with a Single Mobile Sink (MNL–SMS)

Differing from the usual network flow problems that involve multiple s-t flows, MNL–SMS combines two types of problems, namely *maximum concurrent flow* problem and *maximum multicommodity flow* problem. It is a maximum concurrent flow problem because for each demand λ_i associated with node i, we want to find the maximum multiplier T. Meanwhile it is also a maximum multicommodity flow problem, as for each demand t_k, our objective is to maximize $\sum_k t_k$ without caring the particular value of individual t_k. Therefore, we need to design a new algorithm to address it, and our design is based on the proposal of Garg and Könemann [13].

For the case of single mobile sink, $\mathcal{S}$ includes only one sink referred to as s. Clearly, s can choose its location from those indicated by $\mathcal{V}$, implying that $\mathcal{L}_k$ has $|\mathcal{V}|$ possibilities. Therefore, the dimension of $[t_k]$ is at most $|\mathcal{V}|$, and thus we further simplify the problem by assuming that s is coincident with location $k \in \mathcal{V}$ during the kth epoch. Due to the symmetry of the MNL–SMS problem, the order of the sink locations does not affect the optimal solution. In other words, this assumption leads to the solution applicable to a general case without location ordering. We omit the formulation of the MNL–SMS problem, as it is indeed the same as that of the general MNL case (18.12)–(18.19) under the condition that $\mathcal{L}_k = \{k\}$. Similarly, let $W(i,k) = \sum_{j \in \min\left\{p | p \in P_{ik}^k\right\}, j \neq k} w(j) \left(e_j^{\mathrm{T}} + \mathrm{I}_{j \neq i} \cdot e^{\mathrm{R}}\right)$, we get the separation oracle for MNL–SMS as follows:

$$\rho(\mathbf{w}) = \min_k \rho_k(\mathbf{w}) = \min_k \left[\sum_i \lambda_i \left(\sum_{j \in \min\left\{p | p \in P_{ik}^k\right\}, j \neq k} w(j) \left(e_j^{\mathrm{T}} + \mathrm{I}_{j \neq i} \cdot e^{\mathrm{R}}\right) \right) \right] \tag{18.23}$$

Here the oracle computes $\min_k \rho_k(\mathbf{w})$ by first using the Floyd–Warshall algorithm [9] to compute all-pairs shortest path with a time complexity $\Theta(n^3)$. Paths ended at a common node are then clustered into groups, and the algorithm searches for the best "median" k that achieves $\min_k \rho_k(\mathbf{w})$. As the complexity of clustering and searching is negligible compared to the Floyd–Warshall algorithm, this oracle has a complexity of $\Theta(n^3)$. Now, we are ready to derive the algorithm for solving the MNL–SMS problem. The pseudo-code is provided, where superscript $^+$ indicates the updated value, $\delta = (1+\varepsilon)[(1+\varepsilon)n]^{-1/\varepsilon}$, and ε is the required error bound.

Algorithm 1 MNL_ALGO

```
Input: $\mathcal{N}$, $\mathcal{E}$, $\Lambda = [\lambda_i]$, $\mathbf{E} = [E_i]$, $\mathbf{e} = \left[e_i^{\mathrm{T}}\right]$, $e^{\mathrm{R}}$, and initial weight assignment $\mathbf{w} = [\delta / E_i]$, $\forall i \in \mathcal{N}$
1: repeat
2:     Identify the most violated element by the oracle: $k^+ = \arg\min_k \rho_k(\mathbf{w})$;
3:     Increase the $k$th epoch by 1 unit: $t_k^+ = t_k + 1$;
4:     Follow the shortest path $p_{ik}^+$ (suggested by the oracle) from each node $i$ to the sink, route
       $\lambda_i$ units of commodity along that paths, and update the flow through node $j \in p_{ik}^+$:
       $f^+(j) = f(j) + \lambda_i$;
5:     Update the weight of node $i$: $w^+(i) = w(i)\left(1 + \varepsilon f^+(i)\left(e_i^{\mathrm{T}} + \mathrm{I}_{j \neq i} \cdot e^{\mathrm{R}}\right)/E_i\right)$;
6:     Calculate the dual objective:
       $G(\mathbf{w}^+) = \sum_i E_i w^+(i) = G(\mathbf{w}) + \varepsilon \sum_i w(i) f^+(i)\left(e_i^{\mathrm{R}} + e^{\mathrm{T}}\right) = G(\mathbf{w}) + \varepsilon \cdot \rho(\mathbf{w})$;
7: until $G(\mathbf{w}^+) \geq 1$
8: return maximum network lifetime: $\tilde{T} = \log_{1+\varepsilon}^{-1} \dfrac{1+\varepsilon}{\delta} \sum_k t_k^+$
```

The algorithm proceeds in iterations. In each iteration, the oracle identifies k^+ that gives the minimum "potential," and it also suggests the paths from each node to that sink. Then the weight assignments $\mathbf{w}$, time schedule t_k, flow assignments $[f(i)]$, and dual objective $G(\mathbf{w})$ are all updated accordingly; they will serve as the input to the next round of iteration. The algorithm runs until the dual objective exceeds the threshold 1. We show the correctness and the time complexity of this algorithm by the following proposition.

Proposition 3 *Given* $\sum_i \lambda_i \leq E_i / \left(e_i^{\mathrm{T}} + \mathrm{I}_{j \neq i} \cdot e^{\mathrm{R}}\right), \forall i,$[4] ***MNL_ALGO*** *computes a* $(1-\varepsilon)^{-2}$*-approximation to the MNL–SMS problem in time* $\Theta(n \log n) \cdot T_{\text{oracle}}$*, where* T_{oracle} *is the time complexity for the oracle to compute* $\min_k \rho_k(\mathbf{w})$.

Proof Let the dual optimal value be β. According to the 6th step, we have at the end of each iteration

$$G(\mathbf{w}^+) \leq G(\mathbf{w})(1 + \varepsilon/\beta) \leq G(\mathbf{w})e^{\varepsilon/\beta}$$

where $\dfrac{G(\mathbf{w})}{\rho(\mathbf{w})} \geq \beta$ accounts for the first inequality. Suppose that $G(\mathbf{w}^+) \geq 1$ at the end of the tth iteration and given initially $G(\mathbf{w}) = n\delta$, we have

$$1 \leq G(\mathbf{w}^+) \leq n\delta e^{t\varepsilon/\beta} \Rightarrow \frac{\beta}{t} \leq \frac{\varepsilon}{\ln(n\delta)^{-1}}$$

The ratio between the dual and primal solutions is given by

$$\gamma = \frac{\beta}{\tilde{T}} = \frac{\beta}{t} \log_{1+\varepsilon} \frac{1+\varepsilon}{\delta} \leq \frac{\varepsilon \log_{1+\varepsilon} \dfrac{1+\varepsilon}{\delta}}{\ln(n\delta)^{-1}} = \frac{\varepsilon}{\ln(1+\varepsilon)} \cdot \frac{\ln \dfrac{1+\varepsilon}{\delta}}{\ln(n\delta)^{-1}}$$

As $\delta = (1+\varepsilon)[(1+\varepsilon)n]^{-1/\varepsilon}$, we have

$$\gamma \leq \frac{\varepsilon}{(1-\varepsilon)\ln(1+\varepsilon)} \leq \frac{\varepsilon}{(1-\varepsilon)(\varepsilon - \varepsilon^2/2)} < (1-\varepsilon)^{-2}$$

As the maximal lifetime $\hat{T} = \beta$ due to strong duality, we have $\tilde{T} = t \cdot \log_{1+\varepsilon}^{-1} \dfrac{1+\varepsilon}{\delta} > (1-\varepsilon)^2 \hat{T} \geq (1-2\varepsilon)\hat{T}$. Q.E.D.

We only sketch the proof here but omit the detailed proof for feasibility and time complexity. Interested readers are referred to [22] for details.

18.2.4.2 Approximation Algorithm for General MNL

As we pointed out in Sect. 18.2.2, if we had an oracle to solve the p-median problem, then we would be able to solve the general MNL problem under the on-graph mobility scenario, because the dual LP (18.17) and (18.18) serves as a polynomial-time reduction from the separation oracle (a p-median problem) to the general MNL problem. For the off-graph mobility, we could extend the graph $\mathcal{G}$ by assigning zero

[4] This assumption is reasonable because each sensor node should be equipped with an energy source that is at least enough for the node to forward data for all nodes in one time unit. Otherwise if a node $i : E_i / \left(e_i^{\mathrm{T}} + \mathrm{I}_{j \neq i} \cdot e^{\mathrm{R}}\right) < \sum_i \lambda_i$ is deployed close to a static sink (assuming a randomly deployed WSN), the network lifetime can be even less than one time unit. In addition, it can be proved that an approximation ratio of $(1-\varepsilon)^{-3}$ is still achievable without this assumption.

weight to every vertex in $\mathcal{V} \backslash \mathcal{N}$ and connecting it to every other vertex through a directed edge. This extension allows us to run the p-median solver on $\mathcal{G}$ without being interfered by the vertices representing those potential off-graph sink locations.

However, unless P = NP, no efficient p-median solver would exist. Although there exist approximation algorithms for the p-median problem, there is no guarantee that the dual LP may accommodate an approximate oracle. Fortunately, the algorithm we proposed in Sect. 18.2.4.1, MNL_ALGO, does accommodate an approximate oracle, with a slight change in the 2nd step as $\mathcal{L}_{k^+} : k^+ = \arg\min_k \rho_k(\mathbf{w})$. Therefore, combining any PTAS for p-median with MNL_ALGO will yield a PTAS for MNL, as shown by the following proposition.

Proposition 4 *If the p-median oracle can be approximated within a ratio of $\alpha > 1$ (i.e., the oracle has an α-approximation), then* ***MNL_ALGO*** *along with this oracle provides an $\alpha \cdot (1-\varepsilon)^{-2}$-approximation to the general MNL problem.*

Proof The main difference between MNL and MNL–SMS is that, instead of having an oracle that returns the exact $\rho(\mathbf{w})$, we only have an α-approximation of the oracle. It means that the oracle always returns $\tilde{\rho}(\mathbf{w}) \leq \alpha\rho(\mathbf{w})$ with $\alpha > 1$. Since we have $\dfrac{G(\mathbf{w})}{\rho(\mathbf{w})} > \beta$ for MNL–SMS, we now have $\dfrac{G(\mathbf{w})}{\tilde{\rho}(\mathbf{w})} > \tilde{\beta}$, where $\tilde{\beta} = \dfrac{\beta}{\alpha}$. Therefore, we basically follow the line of proving **Proposition 3** but replacing β by $\tilde{\beta}$, and we will finally have $\tilde{T} > \dfrac{(1-\varepsilon)^2}{\alpha}\hat{T}$. Q.E.D.

In fact, Arya et al. [2] gave a $(3+\omega)$-approximation algorithm for the p-median problem. Therefore, we have an algorithm to approximate the general MNL problem with a factor of $(3+\omega)(1-\varepsilon)^{-2}$.

18.2.5 Numerical Results

In this section, we show the quantitative improvement on lifetime by using a mobile sink for WSNs. We always assign a homogeneous λ, e^{T}, e^{R}, and E to all nodes in order to facilitate the interpretation of the results. Without loss of generality, we assume $\lambda = 1$, $e^{\mathrm{T}} = e^{\mathrm{R}} = 0.5$, and $E = |\mathcal{N}| = n$. We set $\varepsilon = 0.01$. Here we only investigate two metrics, namely lifetime and pause time distribution, and refer to [23] for the evaluation of other metrics. We only focus on a single mobile sink as we surely have optimal solution for this case. Note that the pause time distribution given by an approximate solution may differ a lot from the optimal solution. The numerical results are obtained for networks with regular and arbitrary topologies. We consider both on-graph and off-graph sink mobilities and compare them for all networks. All these problems are solved using the primal-dual algorithm presented in Sect. 18.2.4.1.

18.2.5.1 Grid Network

For grid networks on $\sqrt{n} \times \sqrt{n}$ lattices, the maximum achievable lifetime by a static sink is $n/(\lceil (n-5)/4 \rceil + 1)$, because the lifetime is maximized if the forwarding load is balanced among the four neighbors of the sink. This lifetime can be obtained by putting the sink at the network center (if $\sqrt{n}$ is odd) or at any of the four nodes close to the center (if $\sqrt{n}$ is even). While this lifetime is converging to 4 when $n \rightarrow \infty$, the lifetime achieved by a mobile sink increases dramatically with the network size (Table 18.2). For small-size networks (e.g., $|\mathcal{V}| = 9$ in Table 18.2), the substitution effect dominates the load balancing effect, so the relative improvement is small. With an increasing network size, the number of alternative paths between an s–t pair is also increasing. Consequently, the load balancing effect becomes increasingly remarkable and thus produces significant improvement on the lifetime.

We illustrate the pause time distribution in four networks in Fig. 18.4. Our observation is that the sink tends to move toward the periphery of a network with an increasing n. The intuition is that, for a 3D grid on a sphere, the sink should pause everywhere with the same time period. Therefore, the pause times spread out when the network grows in size and thus appears more and more like a sphere grid to the nodes close to the center. This observation also corroborates the result in [21]: the network periphery, as a sink moving trace, is asymptotically optimal. Note that we investigate in [21] the asymptotical case where the node density is large enough to make the necessary radio ranges infinitely small. In that case, the shortest paths between any s–t pair happen to be straight lines.

We also consider the off-graph sink mobility, where the sink can also move to the vertices of another grid that is complementary to the original network, as shown in Fig. 18.5(a). The results show that, for all the networks shown in Table 18.2, off-graph mobility **does not** further improve the lifetime compared with on-graph mobility. In fact, even the pause time distribution remains to be the same after relaxing the on-graph constraint on the sink mobility. This interesting observation shows that, for networks that are well connected, on-graph sink mobility is sufficient to achieve the maximum lifetime.

Table 18.2 Comparing the achievable lifetime between using a mobile sink and a static sink in grid networks of different size

	Network lifetime T		
$\lvert\mathcal{V}\rvert$	Mobile sink	Static sink (optimal)	Improvement (%)
9	5.331	4.500	18.47
16	6.509	4.000	62.72
49	11.09	4.084	171.7
121	17.07	4.033	323.2
144	18.71	4.000	376.8
225	23.29	4.018	479.7
289	26.33	4.014	555.9

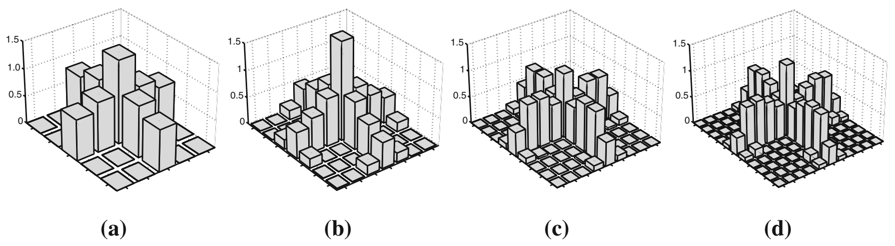


Fig. 18.4 Pause time distribution of a mobile sink in grid networks. The x and y axes indicate the location of the nodes, and the z-axis represents the pause time: (**a**) 25 nodes, (**b**) 49 nodes, (**c**) 81 nodes, (**d**) 121 nodes

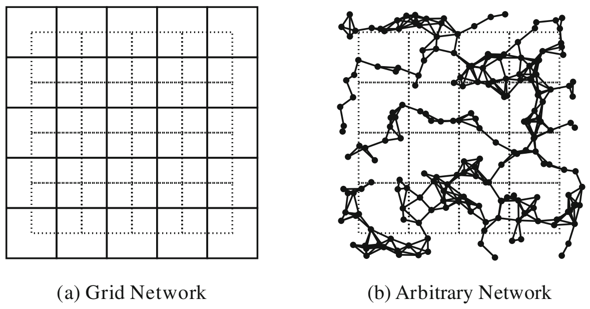


Fig. 18.5 Illustrations of off-graph sink mobility. The original network is represented by (**a**) the solid grid and (**b**) the black nodes. The sink, in addition to being able to move on-graph, may also move to locations represented by the vertices of the dash grid

18.2.5.2 Arbitrary Network

We also perform experiments on arbitrary networks (nodes uniformly distributed within a square). Figure 18.5 (b) shows such a network and the possible off-graph sink locations (represented by the dash grid). We consider both 100-node and 200-node networks with a 10×10 off-graph grid, and each with 30 trials. In Fig. 18.6, we compare the maximum lifetime achieved in four cases, namely, static on-graph sink, mobile on-graph sink, static off-graph sink, and mobile off-graph sink. We use the boxplot to summarize the results we have obtained, in which each case is depicted by five quantities: lower quartile (25%), median, upper quartile (75%), and the two extreme observations. It can immediately be seen that moving the sink always improves the lifetime compared with fixing it, whether on-graph or off-graph. Also, it is not a surprise that allowing off-graph sink locations (for both mobile and static sinks) outperforms constraining those locations on-graph, this is, of course, at a cost of higher complexity in solving the problem. Fortunately, our algorithm handles this complexity very well given a reasonable number of the off-graph locations.

It is also interesting to look at the pause time distribution, Fig. 18.7 illustrates one such case (other cases exhibit the same trend). A direct observation is that the sink tends to pause at the nodes whose degrees are high (for on-graph locations) and at the off-graph locations around which the node density is high. This is intuitive because the more neighbors a node or a location has, the more a balanced load can

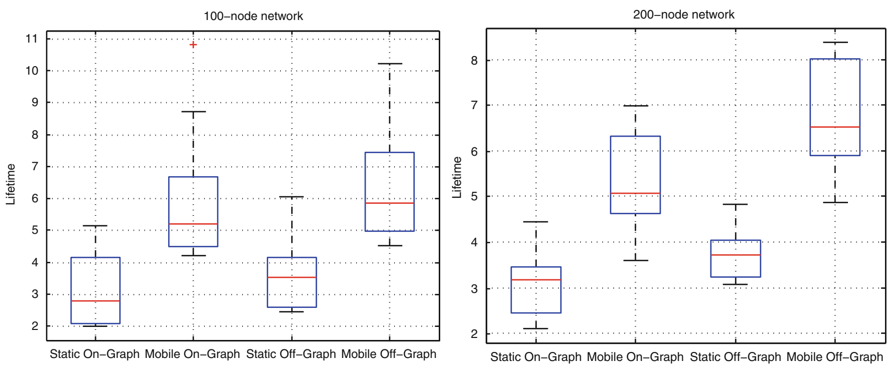


Fig. 18.6 Comparing different sink behaviors in arbitrary networks with 100 and 200 nodes

be achieved by co-locating the sink with it. A slightly surprising observation is that not many locations are chosen by the optimal sink mobility: only 5 positions for on-graph mobility and 10 positions for off-graph mobility. This is quite different from the grid network. In fact, most arbitrarily deployed networks have a topology close to a tree rather than a mesh. It is quite intuitive to see that the sink mobility will concentrate around the root of a balanced tree.

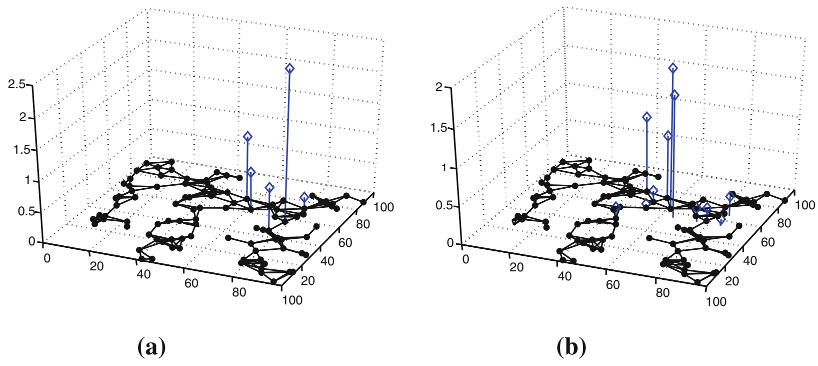


Fig. 18.7 Pause time distribution of a mobile sink in an arbitrary network. The x and y axes indicate the location of the nodes, and the z-axis represents the pause time: (**a**) On-graph mobility and (**b**) Off-graph mobility

18.2.6 Summary

By far, we have formulated the optimization problem for maximizing network lifetime using mobile sinks. We have analyzed the complexity of the problem and characterized the optimal solution through duality theory. Finally, we have developed an algorithm to solve the problem with one mobile sink and to approximate the solution if there are multiple sinks. In the remainder of this chapter, we will present several extensions to the optimization framework defined in this section. In Sect. 18.3, we explain how to obtain optimal solution if we do not put any constraints on the sink locations and hence allow them to be chosen within, for example, a 2D Euclidean space. Recall that the reason accounting for the lifetime improvement in the mobile sink approach is twofold, namely load balancing effect and substitution effect, while the former is the dominating factor. In Sect. 18.4, we will look into another extension where the substitution effect will be fully utilized, which is called the mobile node approach. Finally, we will show, in Sect. 18.5, that certain slight changes of the terminology allow us to model the mobile relay approach. Also, we will describe a variance of the problem formulation, which, by simplifying several assumptions of the model, may actually lead to more efficient solutions.

18.3 Balancing Traffic Load with Mobile Sinks: The Case of Unconstrained Mobility

Although it seems that the problem formulation we present in Sect. 18.2 is limited to the case where a finite set of potential sink locations is provided, we show in this section that it is not true: given a continuous space (e.g., a Euclidean one), we only need to search among a finite number of locations in order to obtain the optimal solution or to closely approximate the optimal solution. In the following, we first discuss, in Sect. 18.3.1, the case where the transmission energy is associated with a node (as assumed in Sect. 18.2.1); the approach we present in Sect. 18.2 still yields the optimal solution even after relaxing the constraints on the potential locations. In Sect. 18.3.2, we slightly change one of the assumptions by associating the transmission cost with a link. The problem does become harder under this circumstance, but we will describe an extension to the approximation scheme presented in [30],[5] which may give a solution that is arbitrarily close to the optimal one for a single mobile sink and provide good approximation for multiple mobile sinks, at an increasing (sometimes drastically) computational complexity.

18.3.1 Node-Associated Transmission Energy

If the transmission energy is associated with individual nodes, we could always come up with a virtual circle around a certain node, such that a link can be established iff the destination node falls within this circle. As shown in Fig. 18.8, the potential locations for the mobile sinks are constrained by these circles. In particular, these circles partition the continuous space into many subareas, whose boundaries are one or more portions of certain circles. Most importantly, we only need one sink

[5] The original work by Shi and Hou [30] is only designed for a single mobile sink. In this section, we extend their approach by combining it with MNL_ALGO presented in Sect. 18.2.4 to deal with multiple mobile sinks.

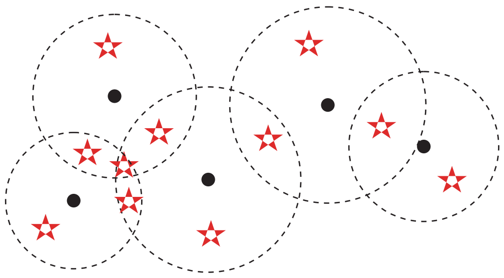


Fig. 18.8 All possible off-graph sink locations (represented by the stars) given a certain assignment of the node-associated transmission energy

location to represent each of these subareas (which can be arbitrarily chosen within a subarea), as the resulting network topology graph remains the same if we move the sink within a subarea. In a nutshell, given a WSN along with its transmission energy assignment and a specific way of defining the mapping **c**, we can compute the radius of the circle for each node. Drawing all these circles will give us a finite set of subareas and hence a finite set $\mathcal{A}$ of potential sink locations. Let $\mathcal{V} = \mathcal{N} \cup \mathcal{A}$, we are back to the off-graph mobility problem addressed in Sect. 18.2.

We consider the model that associates the transmission energy with individual nodes to be more realistic than the link-associated version, because, although nodes may have a tunable transmission power, it is not cost-effective to dynamically tune the power for destinations at different distances. In addition, tuning transmission power according to transmission distances is not always feasible either, as a node might not know the distances. Therefore, a reasonable scenario, in our opinion, is that each node sets up a transmission power according to certain topology control mechanisms [18, 19] at the network initialization phase and fixes this power until some topology changes happen. However, since the link-associated model is also popularly used, we will treat the MNL problem under that model in Sect. 18.3.2.

18.3.2 Link-Associated Transmission Energy

Under on-graph mobility, it is pretty straightforward to switch from the node-associated model to the link-associated version: we simply need to replace e_i^{T} with $e_{i,j}^{\mathrm{T}}$.[6] However, allowing off-graph mobility drastically increases the complexity of the problem. As the (link) transmission energy can be tuned freely, any location in a continuous space is virtually unique as it may yield a different transmission energy from some sensor node to that location.

Whereas it is true that finding the optimal solution will incur a tremendous complexity, obtaining good approximation is still possible [30]. To better illustrate the idea, we first reformulate the MNL problem by modifying the constraint (18.14) as follows (constraint (18.13) is not affected by switching to the link-associated

[6] A byproduct of this change is that the cost assignment **c** is not needed anymore, as any link (i, j) is feasible given a sufficiently high transmission energy $e_{i,j}^{\mathrm{T}}$.

version).

$$\sum_{k:i\notin\mathcal{L}_k}\sum_{p\in P_i^k} f(p)\left(\mathrm{I}_{(i,j)\in p, j\notin\mathcal{L}_k}\cdot e_{i,j}^{\mathrm{T}} + \mathrm{I}_{(i,j)\in p, j\in\mathcal{L}_k}\cdot e_{i,j}^{\mathrm{T}} + \mathrm{I}_{p\notin P_{is}^k,\forall s}\cdot e^{\mathrm{R}}\right) \leq E_i \;\; \forall i \tag{18.24}$$

Note that, apart from replacing e_i^{T} with $e_{i,j}^{\mathrm{T}}$, we also split the first term in the parentheses into two parts: while the first part is independent of the sink locations in the current epoch, the second part is not. We call this problem *maximizing network lifetime with link-associated transmission energy* (MNL–LATE). In theory, the possible choices of $e_{i,j}^{\mathrm{T}}$ for arbitrary $i \in \mathcal{N}$ and $j \in \mathcal{L}_k$ are infinite, this makes the searching for an optimal solution enormously hard. Fortunately, it is possible to develop a $(1+\boldsymbol{\kappa})$-approximation (where $\boldsymbol{\kappa}$ is the error bound) for a single mobile sink or a $(1+\boldsymbol{\kappa})(3+\omega)(1-\varepsilon)^{-2}$-approximation for multiple mobile sinks based on the algorithm given in Sect. 18.2. In the following, we first briefly present the basic idea of the approach in Sect. 18.3.2.1, then we describe the algorithm along with propositions that strictly prove its correctness in Sect. 18.3.2.2.

18.3.2.1 Parameterization Using Geometric Sequence

We first parameterize MNL–LATE by taking $e_{i,j}^{\mathrm{T}}$ in the second term of (18.24) as the parameters. Instead of letting each $e_{i,j}^{\mathrm{T}}$ to be any possible real number, we limit the choice to be a sequence of numbers $\mathbf{e}_{i,j} = \left\{e_{i,j;0}^{\mathrm{T}}, e_{i,j;1}^{\mathrm{T}}, \cdots, e_{i,j;h}^{\mathrm{T}}, \cdots\right\}, \forall i \in \mathcal{N}, j \in \mathcal{S}$, where $e_{i,j;0}^{\mathrm{T}} = a$ and $e_{i,j;h}^{\mathrm{T}} = a(1+\boldsymbol{\kappa})^h$. Obviously, this sequence is a geometric sequence with factor a and common ratio $(1+\boldsymbol{\kappa})$. Assume that there is a set $\mathcal{A}$ of the off-graph locations such that, for each $j \in \mathcal{A}$, we have $e_{i,j}^{\mathrm{T}} \in \mathbf{e}_{i,j}, \forall i \in \mathcal{N}$ and for each $j : e_{i,j}^{\mathrm{T}} \in \mathbf{e}_{i,j}, i \in \mathcal{N}$, we have $j \in \mathcal{A}$. In other words, $\mathcal{A}$ enumerates any possible off-graph location j whose incurred transmission energy from any node i to itself is given in $\mathbf{e}_{i,j}$. If we could also make $\mathcal{A}$ to be finite, then letting $\mathcal{V} = \mathcal{N} \cup \mathcal{A}$ would bring us back to the off-graph mobility problem addressed in Sect. 18.2. We call this problem MNL–DLATE. Of course, the solution to MNL–DLATE may not be optimal to MNL–LATE, as $\mathcal{A}$ does not include all possible off-graph locations. However, it is intuitive to see that the solution should not be far from optimal if the granularity $(1+\boldsymbol{\kappa})$ we use to discretize the continuous space is sufficiently small.

It is worth noting that the aforementioned discrete parametrization applies to many optimization problems, especially those with linear constraints. In Sect. 18.3.2.2, we first show that this discretizaton is possible within a Euclidean space and $\mathcal{A}$ can indeed be made finite. Then we show that the solution to MNL–DLATE is a good approximation to that of MNL–LATE and we also give the approximation ratio. We note that, if the error bound $\boldsymbol{\kappa}$, thus the common ratio $(1+\boldsymbol{\kappa})$, is small , the cardinality of $\mathcal{A}$ (hence that of $\mathcal{V}$) becomes huge, which makes the problem much harder to solve than the node-associated version presented in Sect. 18.3.1.

18.3.2.2 Algorithm Implementation

Based on what is described in Sect. 18.3.2.1, a practical algorithm to approximate the solution of MNL–LATE needs three components: (1) A finite set $\mathcal{A}$ defined by a set of geometric sequences $\{\mathbf{e}_{i,j}\}_{i\in\mathcal{N},j\in\mathcal{A}}$, (2) The algorithm MNL_ALGO we have presented in Sect. 18.2.4, and (3) A proof to show the approximation ratio. Since the second component is ready, we present in the following the procedures that justify both the first and the third components, assuming a Euclidean space and a link transmission power assignment that is strictly increasing in the Euclidean distance from the source to the destination.

As we assume that $e^{\mathrm{T}}_{i,j}$ is an increasing function of $d(i,\ j)$, the Euclidean distance between node i and node j, the sequence $\mathbf{e}^{\mathrm{T}}_{i,j}$ can be generated by properly drawing concentric circles around i and let $e^{\mathrm{T}}_{i,j;h}$ be the transmission energy to reach from i to the hth circle, as shown in Fig. 18.9. Although to reach any location between the $(h-1)$th and the hth circles only requires a transmission energy in the interval $\left(e^{\mathrm{T}}_{i,j;h-1},\ e^{\mathrm{T}}_{i,j;h}\right)$, we deliberately amplify it to $e^{\mathrm{T}}_{i,j;h}$. According to (18.24), such an amplification tightens the constraint. Therefore, the optimal solution to this special version of MNL–DLATE, called MNL–DALATE, is bounded to be feasible under the general MNL–DLATE where each location is associated with the actual transmission energy derived from the exact distance. Consequently, the optimal solution to MNL–DLATE is no smaller than that of MNL–DALATE. This is summarized by the following proposition.

Proposition 5 *If the solution to MNL–DALATE is an $\boldsymbol{\alpha}$-approximation of the solution to MNL-LATE, where $\boldsymbol{\alpha}>1$, the approximation ratio given by the solution to MNL-DLATE is at most $\boldsymbol{\alpha}$.*

The finiteness of $\mathcal{A}$ is the direct consequence of a bounded search region and of the increasing radius of the circles that generate the geometric sequences, and the boundedness of the search region is shown by the following proposition.

Proposition 6 *An optimal solution of MNL–LATE always has its sink locations constrained within the smallest enclosing disk (SED) that contains $\mathcal{N}$.*

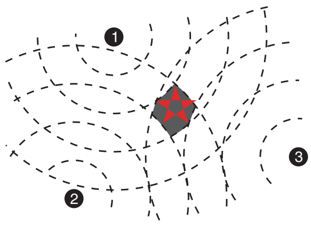


Fig. 18.9 Generating geometric sequences using *concentric circles*. According to our definition of MNL-DALATE, any potential sink location j within the *shaded* region is assigned with a transmission energy vector $\left(e^{\mathrm{T}}_{1,j;2}, e^{\mathrm{T}}_{2,j;3}, e^{\mathrm{T}}_{3,j;3}\right)$

Proof Assume that the statement in the proposition is not true, then there exists at least one optimal sink location that is outside of the SED. We first draw a line segment from this location to the center of the SED, which yields a (point) intersection with the boundary of the SED. Then we mirror the location with respect to the tangent (of the boundary of the SED) that goes through the intersection point. It can be easily seen that this new location is superior to the old one, as it is closer to all the nodes in terms of Euclidean distance. This outcome contradicts the optimality of the assumed "out of SED" location, and hence proves the proposition. Q.E.D.

Now, the algorithm construction becomes pretty clear; we illustrate the algorithm MNL–LATE_ALGO as follows. Note that, if there is only one mobile sink, we can also replace the 4th step by an LP, which removes the (albeit negligible) approximation ratio $(1-\varepsilon)^{-2}$. The performance of MNL–LATE_ALGO is shown by the following propositions.

Algorithm 2 MNL–LATE_ALGO

```
Input: $\mathcal{N}, \Lambda, \mathbf{E}$
1: Compute the SED that contains $\mathcal{N}$, and generate $e^{\mathrm{T}}_{i,j}$ for all $i, j \in \mathcal{N}$;
2: Generate the geometric sequence $\mathbf{e}^{\mathrm{T}}_{i,j}$ for all $i \in \mathcal{N}$ using the sequence of concentric circles,
   and produce the location set $\mathcal{A}$, within the boundary of SED, according to the subareas demar-
   cated by these circles;
3: Applying the amplification rule to generate $e^{\mathrm{T}}_{i,j}$ for all $i \in \mathcal{N}$, $j \in \mathcal{A}$;
4: Call MNL_ALGO upon the instance $(\mathcal{V}, \mathcal{E}, \Lambda, \mathbf{E}, \mathbf{e})$, where $\mathcal{V} = \mathcal{N} \cup \mathcal{A}$, $\mathcal{E}$ includes all the
   edges among $\mathcal{N}$ and those from $\mathcal{N}$ to $\mathcal{A}$ and $\mathbf{e} = \left[e^{\mathrm{T}}_{i,j}\right]$, and get the return value $\tilde{T}$.
5: return maximum network lifetime: $\tilde{T}$
```

Proposition 7 *With a single mobile sink, the value returned by MNL–LATE_ALGO, $\tilde{T}$, is a $(1+\kappa)$-approximation to the optimal value $\hat{T}$ of MNL–LATE, in other words, $\tilde{T} \geq (1+\kappa)^{-1}\hat{T}$.*

Proof The proof goes very similar to the discussion made before **Proposition 5**. We first notice that any location that belongs to an optimal solution of MNL–LATE must fall into one of the subareas demarcated by the concentric circles and the SED. Then it is straightforward to see that the amplification rule simply exaggerates the transmission energy incurred by an optimal location, which, in effect, tightens the constraint (18.24). The other important fact is that the amplification is bounded: it is at most $(1+\kappa)$ to the value that would have been incurred by an optimal location. Given $\hat{T}$ as the optimal value of MNL–LATE, we can scale it down by a factor $(1+\kappa)$, which effectively scales down all $f(p)$'s in (18.24), and makes it a feasible solution to MNL–DALATE. As $\tilde{T}$ is the optimal value of MNL–DALATE (assume the use of LP for the 4th step), we have $\tilde{T} \geq (1+\kappa)^{-1}\hat{T}$. Q.E.D.

Similar arguments lead us to the approximation ratio for multiple mobile sinks.

Proposition 8 *With multiple mobile sinks, $\tilde{T}$ is a $(1+\kappa)(3+\omega)(1-\varepsilon)^{-2}$-approximation to the optimal value $\hat{T}$ of MNL–LATE.*

Note that the amplification procedure is simply used to facilitate the proof of approximation ratio. In a practical implementation of MNL–LATE_ALGO, we may skip it and simply choose an arbitrary location within a particular subarea to represent that subarea. In other words, we solve MNL–DLATE rather than MNL–DALATE. The solution, though bearing the same worst-case performance as the one with amplification, is in general better, i.e., closer to the optimal value.

18.3.3 Summary

We demonstrate in this section that though the formulation and solution presented in Sect. 18.2 have a constrained set of locations for mobile sinks to choose, it is not difficult, in theory, to extend them by relaxing the constraint. However, we believe that all the results presented in the section are more for pure theoretical purpose, as the improvement (in terms of the absolute value of the lifetime) can be very marginal and the cost to obtain this improvement is huge. Especially in the link associated case, the cardinality of $\mathcal{A}$ is in the order of $\left(\frac{n}{\kappa}\right)^2$, which can be enormous if we are chasing an accurate approximation with very small κ. Therefore, while appreciating the beauty of the theory, we do caution the readers for any practical use of it.

18.4 Energy Conservation with Mobile Nodes: The Extreme Usage of the Substitution Effect

Although the substitution effect (see Sects. 18.2.1 and 18.2.3) is overwhelmed by the load balancing effect in the mobile sink approach, one might still wonder if it is possible to fully exert the benefit of this effect. As illustrated by Fig. 18.10, such an approach does exist [35], where the sink is kept static while some powerful mobile nodes are changing their location from time to time to replace certain overloaded (static) nodes. However, the analysis performed in [35] is based on a fluid model (similar to [21]), hence the results hold only in an asymptotic sense and cannot be

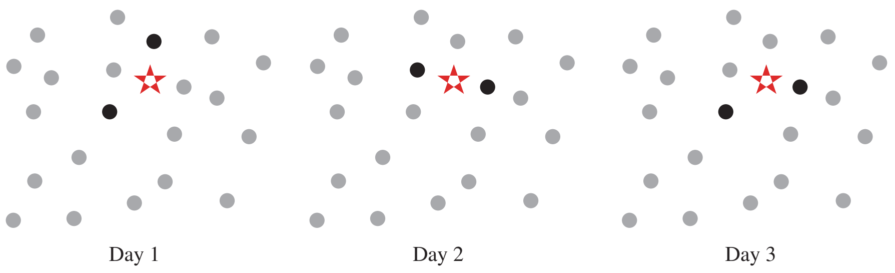


Fig. 18.10 Using mobile nodes to balance the traffic load within a WSN. The star represents the static sink, and nodes with darker color are the mobile nodes. Note that the mobility is (again) slow, as the mobile nodes may change their locations very infrequently

applied whenever a specific network topology is given. Therefore, we reformulate the problem following the general framework presented in Sect. 18.2. The great benefit is that, as we will show later, almost all the results presented in Sect. 18.2 apply here, only with a change of the separation oracle.

18.4.1 MNL with Multiple Mobile Nodes (MNL–MMN)

We keep using the model and terminologies presented in Sect. 18.2.1. The differences are (1) there is only one static sink $s \in \mathcal{N}$ and (2) $\mathcal{L}_k \in \mathcal{N}$ denotes the mobile node (rather than sink) locations during the kth epoch. Note that, as the substitution effect demands co-locations of mobile nodes with certain sensor nodes, the problem only has an on-graph version, i.e., $\mathcal{V} = \mathcal{N}$, where $\mathcal{V}$ represents potential locations of the mobile nodes. Let us directly formulate the MNL–MMN problem into its Path-Flow form:

$$\text{maximize} \quad T = \sum_k t_k \tag{18.25}$$

$$\text{subject to} \quad \sum_{p \in P_{is}^k} f(p) \geq \lambda_i t_k \quad \forall i \neq s, k \tag{18.26}$$

$$\sum_{k: i \notin \mathcal{L}_k} \sum_{p \in P_i^k} f(p) \left(e_i^{\mathrm{T}} + \mathrm{I}_{p \notin P_{is}^k} \cdot e^{\mathrm{R}} \right) \leq E_i \quad \forall i \neq s \tag{18.27}$$

$$t_k, f(p) \geq 0 \quad \forall k, p \tag{18.28}$$

where I_A is the indicator function of event A, p is a path between a node and the sink s, and $f(p)$ is the flow going through that path. Furthermore, we denote by P_{is}^k the path set from node i to the sink s and by P_i^k the set of paths going through node i, both in the kth epoch. Splitting a flow among a set of paths implies that we allow multi-path routing strategy to be taken by the optimal solution. As this formulation is very similar to that of MNL (18.13) and (18.14), the same happens to the dual problem of MNL–MMN.

$$\text{minimize} \quad G(\mathbf{w}) = \sum_i E_i w(i) \tag{18.29}$$

$$\text{subject to} \quad \sum_{i \neq s} \lambda_i W(i,k) \geq 1 \quad \forall k \tag{18.30}$$

$$\sum_{j \in p \in P_{is}^k, j \notin \mathcal{L}_k, j \neq s} w(j) \left(e_j^{\mathrm{T}} + \mathrm{I}_{j \neq i} \cdot e^{\mathrm{R}} \right) - W(i,k) \geq 0 \quad \forall i \neq s, k, p \tag{18.31}$$

$$w(i), W(i,k) \geq 0 \quad \forall i \neq s, k \tag{18.32}$$

where the $w(i)$ is the weight assigned to node i, representing the marginal cost of using an additional unit energy of node i; and $W(i,k)$ is the weight of a commodity, i.e., data flow going from node i to sink s during epoch k; it indicates the marginal cost of rejecting a unit demand of the commodity. The striking similarity between MNL and MNL–MMN immediately suggests the validity of applying most of the results obtained in Sect. 18.2 to MNL–MMN, with, of course, certain exceptions, as we will discuss in Sect. 18.4.2.

18.4.2 Theorem, Complexity, and Algorithm

We directly use the following theorem to characterize the optimal solution of MNL–MMN; detailed analysis is omitted as it basically follows the same line as in Sects. 18.2.2 and 18.2.3.

Theorem 3 (MAX-LIFETIME MIN-POTENTIAL RATIO THEOREM RELOADED) *Given the lifetime maximization problem formulated in (18.25, 18.26, 18.27, 18.28), the optimal lifetime $\hat{T}$ is such that*

$$\hat{T} = \min_{\mathbf{w}} \left[\frac{G(\mathbf{w})}{\rho(\mathbf{w})} \right]$$

where $G(\mathbf{w}) = \sum_i E_i w(i)$ is a linear combination of the energy reserves of all nodes with coefficients $w(i)$, and

$$\rho(\mathbf{w}) \equiv \min_k \rho_k(\mathbf{w}) = \min_k \left[\sum_i \lambda_i \left(\sum_{j \in \min\{p | p \in P_{is}^k\}, j \notin \mathcal{L}_k, j \neq s} w(j) \left(e_j^{\mathrm{T}} + \mathrm{I}_{j \neq i} \cdot e^{\mathrm{R}} \right) \right) \right]$$

is the minimum "potential" (computed as the sum of the minimum "cost," given $w(i)$, to route λ_i from node i to the sink s) achieved among all possible mobile node layouts $\{\mathcal{L}_k\}$.

Similar to the on-graph MNL, the answer to TMNTM is also positive for **MNL–MMN**. We omit the proof here; it follows the same line as the proof for **Proposition 2**. Note that the question asked by TMNTM for MNL–MMN is whether moving the mobile nodes is superior to keeping them static. Moreover, for certain cases where (off-graph) sink mobility does not improve network lifetime, adding (on-graph) mobile nodes may still benefit network lifetime. Taking the networks shown in Fig. 18.3 as examples, moving some mobile nodes (even just one) around the rings to replace those static nodes in turn is bounded to improve the network lifetime. The latter fact seems to suggest that it would be better to combine mobile nodes and mobile sinks.

Note that $\rho(\mathbf{w})$ is also the separation oracle of the dual MNL-MMN. Let $K = |\mathcal{L}_k| = m$, $\boldsymbol{\omega}(i) = \boldsymbol{\lambda}_i$, $\ell(j) = w(j) \left(e_j^{\mathrm{T}} + \mathrm{I}_{j \neq i} \cdot e^{\mathrm{R}} \right)$, and $d(i) =$

$\sum_{j\in\min\{p|p\in P_{is}^k\},j\neq s}\ell(j)$, then the separation oracle is equivalent to the following decision problem:

INSTANCE: A graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, a weight assignment $\omega(i) : \mathcal{N} \to \mathbb{R}_0^+$, a length assignment $\ell(i) : \mathcal{N} \to \mathbb{R}_0^+$, positive integer $K \leq |\mathcal{N}|$, a special vertex $s \in \mathcal{N}$, and positive real number B.

QUESTION: Is there a set $\mathcal{P}$ of K points on $\mathcal{G}$ such that, if we set $\ell(i) = 0, i \in \mathcal{P}$ and let $d(i)$ be the length of the shortest path from i to s, then $\sum_i \omega(i) \cdot d(i) \leq B$?

If we could solve this decision problem or give a proper approximation to its optimization version, we would be able to apply MNL_ALGO introduced in Sect. 18.2.4 to solve MNL–MMN. However, we show in the following that the separation oracle of dual MNL–MMN is NP-hard, and hence MNL–MMN is also NP-hard, again due to [26] (Theorem 3).

Proposition 9 *The separation oracle of MNL–MMN is NP-hard.*

Proof The NP-hardness of the separation oracle can be shown by transforming from the p-median problem on special cases such as the one shown in Fig. 18.11. It is straightforward to see that $\mathcal{P}$ has to be chosen among $\mathcal{V}_2$, in order to get a maximum weight reduction $M(K + |\mathcal{V}_1|)$ in $\sum_i \omega(i) \cdot d(i)$. However, which $\mathcal{P}$ vertices to be chosen among $\mathcal{V}_2$ are determined by the solution of the p-median problem on $\tilde{\mathcal{G}}(\mathcal{V}_1, \mathcal{E}_1)$, where $\mathcal{E}_1$ refers to the set of edges whose both ends are in $\mathcal{V}_1$. This is so because the answer to the separation oracle is true iff the answer to the p-median problem with $B' = B - M(|\mathcal{V}_2| - K)$ is true. Conversely, if we could address the separation oracle for $\mathcal{G}(\mathcal{V}_1 \cup \mathcal{V}_2 \cup \{s\}, \mathcal{E})$, we would actually solve the p-median problem in $\tilde{\mathcal{G}}(\mathcal{V}_1, \mathcal{E}_1)$, as the medians are indicated by the chosen $\mathcal{P} \subset \mathcal{V}_2$. Q.E.D.

The existence of efficient PTAS with provable approximation ratio to this separation oracle, to our best knowledge, is unfortunately unknown, although heuristics with good empirical performance can be derived based on short-path algorithms.

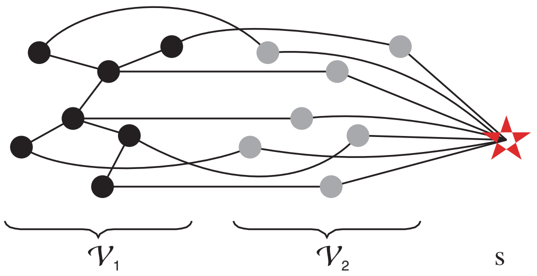


Fig. 18.11 A graph $\mathcal{G}(\mathcal{V}_1 \cup \mathcal{V}_2 \cup \{s\}, \mathcal{E})$, where vertices in $\mathcal{V}_1$ are connected to s only through some vertices in $\mathcal{V}_2$, and vertices in $\mathcal{V}_2$ are not directly connected to each other. We assign uniform $\omega(i) = 1$ to all vertices in both $\mathcal{V}_1$ and $\mathcal{V}_2$, and we assign $\ell(i) = 1$ to vertices in $\mathcal{V}_1$ and $\ell(i) = M$ to vertices in $\mathcal{V}_2$, where $M > |\mathcal{V}_1|$

Therefore, we restrict ourselves to the single mobile node case when making comparisons between the mobile node and mobile sink approach in Sect. 18.4.3.

18.4.3 Numerical Results

We apply the same settings as those in Sect. 18.2.5, and we compare the two approaches, namely mobile sink and mobile node, for both grid networks of different sizes and arbitrary networks with 100 nodes. We illustrate these comparisons in Fig. 18.12. Figure 18.12a shows that whereas the improvement (against the static sink approach) brought by the mobile node approach is more or less a constant, the mobile sink approach yields an increasing improvement in larger networks. The better performance for the mobile node approach in small networks is not a surprise: the substitution effect is the dominating factor of the lifetime improvement in these networks, as already explained in Sect. 18.2.5. In Fig. 18.12b, we use MS, SS, and MN to refer to mobile sink, static sink, and mobile node, respectively. When the mobile node approach is used, the static sink is always put at its optimal location. Although the mobile node approach is still inferior to the mobile sink approach in most cases, the difference is much less significant than the cases of grid networks.

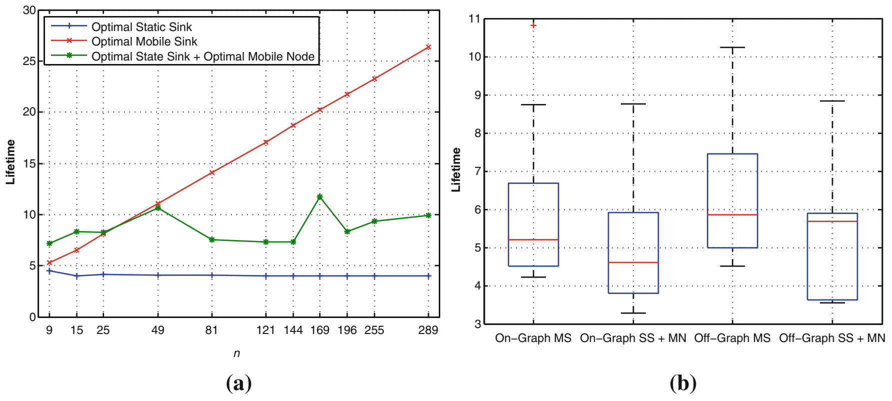


Fig. 18.12 Comparing mobile node with mobile sink: (**a**) Grid networks of different size and (**b**) Arbitrary Networks with 100 nodes

18.4.4 Summary

In this section, we have shown the close relation between the mobile sink and mobile node approaches, by unifying their problem formulations and solution techniques. The other benefit of formulating the mobile node approach into our general optimization framework is to allow detailed investigations on particular network topologies. This actually brings us a slight surprise: Although the mobile sink approach appears to be superior to the mobile node approach in regular topologies (or their asymptotic version [35]), their performances are not very different from each other in arbitrary topologies.

18.5 Energy Conservation with Mobile Relays: Using Mechanical Data Transportation Smartly

Although running WSNs under the slow mobility regime (such as the mobile sink and mobile node approaches presented in the previous sections) may significantly improve the network lifetime, the need for dynamic routing configurations according to the specific locations of the mobile entities could incur additional maintenance overhead in practice. An alternative solution is to use mobile relays to "pick up" data from node through one-hop transmissions and then to transport the data with mechanical movements [5, 28]. Unfortunately, this extreme approach incurs a large transmission delay due to the limited speed of mechanical movements. In this section, we discuss a good compromise made between the aforementioned two extremes: A hybrid approach that jointly considers multi-hop transmissions and mechanical data transportation [32]. As shown in Fig. 18.13, certain locations in the Euclidean plane where the considered WSN locates are chosen as *rendezvous points* (RPs) [36]. A mobile relay periodically travels along a predefined tour and picks up data at RPs, while nodes that may directly reach a RP buffer (or even aggregate[7]) data originated from other nodes and transfer the data to the relay when it arrives at the RP. The advantage of this approach, compared with those slow mobility approaches, is that the multi-hop routing can be configured offline, as it does not change with different locations of the mobile relays.

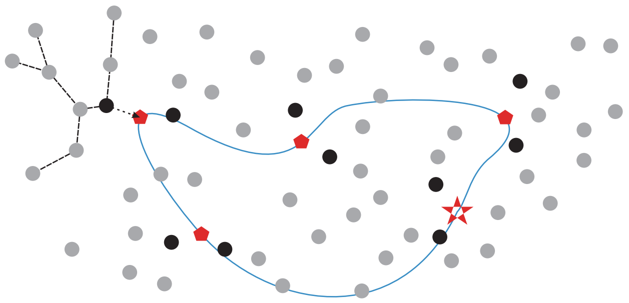


Fig. 18.13 Using a mobile relay to reduce the energy consumption of sensor nodes. The star represents the relay, the pentagons are its RPs, and nodes with darker color are those that buffer or aggregate data sent from other nodes. In other words, a darker colored node is the root of one data collection tree; one of such trees is illustrated at the upper-left corner. The mobility is fast in this case, as the mobile relay is usually required to finish the tour through all RPs within a given time period

[7] By aggregation, we refer to any transformation that summarizes or compresses the data acquired and received by a certain node and hence reduce the volume of the data to be sent out, e.g., [25].

18.5.1 The Single Mobile Relay Positioning (SMRP) Problem

We discuss the problem of identifying the RPs for one mobile relay, which we term *single mobile relay position* (SMRP) problem. Our discussion will be based on the general framework introduced in Sect. 18.2.1. This means that there is a set of potential locations $\mathcal{V}$ within which we would like to identify the optimal subset $\mathcal{S}$ as the RPs. The relay mobility is on-graph if $\mathcal{V} = \mathcal{N}$ or is off-graph if $\mathcal{V} \supset \mathcal{N}$. Usually, $|\mathcal{S}|$ is bounded by a certain integer, as the relay can only visit up to that number of locations given a certain data delivery deadline. Also, we assume there is no data aggregation involved in the routing configuration. The reason we focus only on a single mobile relay is twofold: first, it is already very hard to solve the problem involving only a single relay, and second, solutions for multiple mobile relays could always be based on those for a single mobile relay.

To simplify the exposition, we present the problem formulation as a decision problem rather than an optimization problem. It is well known that these two formulations are equivalent as there always exists a polynomial-time reduction from one to the other.[8] We present the problem for on-graph mobility first and then show how to extend it to off-graph mobility.

INSTANCE: A set of nodes $\mathcal{N}$, a cost assignment $\mathbf{c} : c(i, j) = e, \forall i, j \in \mathcal{N}$, a set $\mathcal{S}$ of *virtual sink location*s with $|\mathcal{S}| < |\mathcal{N}|$, and for each $i \in \mathcal{N}$, a transmission energy e_i^{T}, a receiving energy e^{R}, an energy reserve E_i, a rate λ_i, a constraint that i sends data to only one $s \in \mathcal{S}$, and a positive real number t.

QUESTION: Is there a *rendezvous schedule* $\{\delta_{is}\}$, where $\delta_{is} : \mathcal{N} \times \mathcal{S} \to \{0, 1\}$, $\sum_i \delta_{is} = 1$ (only one mobile relay is allowed) and $\sum_i \sum_s \delta_{is} \le |\mathcal{S}|$, such that the lifetime T is at least t?

In order to extend the formulation to accommodate off-graph mobility, we simply need to assign a zero rate to those vertices in $\mathcal{V} \backslash \mathcal{N}$. It is obvious that the nodes that take a higher traffic load are those that are one-hop from one of the RPs (those darker colored ones shown in Fig. 18.13). Therefore, the objective of the optimization problem is, again, to minimize the maximum load, or, in other words, to balance the traffic load. If we assume the same transmission energy e^{T} for all nodes, the bottleneck node that constrains the lifetime is the one that serves as the root of the "heaviest" data collection tree, where the weight of a tree is the total traffic load generated by all the nodes in the tree. It is straightforward to see that SMRP is

[8] As an example, the decision problem related to MNL (on-graph) is the following:

INSTANCE: A set of nodes $\mathcal{N}$, a cost assignment $\mathbf{c} : c(i, j) = e, \forall i, j \in \mathcal{N}$, a set $\mathcal{S}$ of sinks with $|\mathcal{S}| < |\mathcal{N}|$, and for each $i \in \mathcal{N}$, a transmission energy e_i^{T}, a receiving energy e^{R}, an energy reserve E_i, a rate λ_i, and a positive real number t.

QUESTION: Is there a *sink layout schedule* $\{(sl_k, t_k)\}$ (sl_k is a vector of $[\delta_{is}^k]$ where $\delta_{is}^k : \mathcal{N} \times \mathcal{S} \to \{0, 1\}$ and $\sum_i \sum_s \delta_{is}^k = |\mathcal{S}|$) such that the lifetime $T = \sum_k t_k$ is at least t?

This problem can be shown as NP-hard by, for example, a polynomial-time reduction from the DOMINATING SET on a *unit disk graph* [22].

equivalent to the notoriously hard *base station placement* (BSP) problem [4], which has been shown as NP-hard and whose PTAS is not known by far.[9]

18.5.2 A Variation of SMRP

The difficulty of the problem formulated in Sect. 18.5.1 stems from the fact that it is the property of individual data collection trees that needs to be optimized. In a recent proposal, Xing et al. [36] suggest a simplified problem formulation by (1) assuming a uniform data rate from all nodes that produce sensory data (or source nodes),[10] (2) identical transmission energy and zero receiving energy for all nodes, (3) allowing "many-to-one" data aggregation, and (4) optimizing total energy consumption instead of lifetime. While the first three lead to identical traffic load on each node,[11] the last change shifts the objective from individual data collection trees to the WSN as a whole. The direct consequence of these simplifications is the following:

- The problem formulation may explicitly involve the data delay factor as a design constraint. This eventually translates to the constraint on the length of the tour through all RPs, which appears to be at least as hard as the EUCLIDEAN TRAVELING SALESMAN (ETS) problem.
- The problem objective becomes minimizing the number of links that are needed to connect all the source nodes to the RPs. Due to the identical transmission energy for every node, the energy consumption of transmitting data along a routing path can be approximated by the Euclidean distance between the source and the destination, which in turn suggests an approximation of the objective by the Euclidean length of all routing trees. This approximated objective is actually a GEOMETRIC STEINER TREE (GST) problem.

Although both ETS and GST are NP-complete, ETS actually has a straightforward 2-approximation algorithm given by GST, while GST admits a PTAS whose ratio is pretty close to 1 (can be, in fact, smaller than $\frac{2}{\sqrt{3}}$ [8]). Based on these observations, efficient approximation algorithms can be designed to identify RPs within an unconstrained Euclidean space, unlike SMRP whose RPs are limited to $\mathcal{V}$. In the original proposal [36], two algorithms are given, respectively, for choosing RPs in 2D (e.g., the region covered by a WSN) and 1D (e.g., a fixed track) spaces. We only discuss the first algorithm in this section.

[9] Although Shi et al. [31] have proposed an approximation algorithm for BSP, using a technique similar to the one presented in Sect. 18.3.2.1 that algorithm is only pesudo-polynomial in time, as the time complexity is actually exponential in the number of base stations.

[10] This is a special case of our general formulation, which assigns a uniform data rate to the source nodes and a zero rate to others.

[11] The "many-to-one" data aggregation implies that, no matter how many unit of flows converge at an intermediate node, node only send one unit flow out. These are cases where special aggregation functions such as AVERAGE, MAX, or MIN are used.

Assume the tolerable data delivery delay is D, meaning that the total length of the relay tour must be no more than $L = D\bar{v}$, where $\bar{v}$ is the average speed of the mobile relay. The problem (decision version) can be specified as

INSTANCE: A set of **source** nodes $\mathcal{N}$, two positive real numbers L and C.

QUESTION: Is there a mobile relay tour $\mathcal{U}$ no longer than L and a set of geometric trees $\{\mathcal{T}_k(\mathcal{V}_k, \mathcal{E}_k)\}$ rooted on $\mathcal{U}$ such that 1) $\mathcal{V}_k \subseteq \mathcal{V}$ and $\mathcal{N} \subseteq \bigcup_k \mathcal{V}_k$, and 2) $\sum_k \sum_{(i,j)\in\mathcal{E}_k} d(i, j)$ (where $d(i, j)$ is the Euclidean distance between node i and node j) is no greater than C?

Note that a tree edge $(i, j) : i, j \in \mathcal{V}_k$ does not necessarily represent a physical link; it instead may represent a routing path that goes through other non-source nodes (nodes not in $\mathcal{N}$ but belong to the WSN). Obviously, the optimization version of the problem aims at minimizing the total Euclidean length of all routing paths that are involved in data transmission. This problem can be shown to be NP-hard by a reduction from the ETS problem. Fortunately, due to the reason explained earlier, there exists good approximation algorithms to solve this problem.

To motivate the algorithm, let us first consider an extreme situation where the mobile relay is replaced by a static sink. In this case, the optimal routing tree connecting all source nodes with minimum total length is given by the GEOMETRIC STEINER TREE (GST). Now, we let the sink start moving and thus serve as the mobile relay. As GST provides a lower bound of ETS, moving the relay along the GST appears to be a reasonable choice: as it may strike a good balance between minimizing the transmission cost and limiting the tour length. Based on these observations, the approximation algorithm *rendezvous design for variable tracks* (RD–VT) makes use of approximate GST and ETS solvers as the oracles to address the problem of minimizing $\sum_k \sum_{(i,j)\in\mathcal{E}_k} d(i, j)$ under constrained tour length L. The algorithm first constructs an approximate minimum GST, and then recursively traverses it in depth-first manner to find proper RPs. Initially, the length to be traversed on the GST is set as $L/2$. For each recursion, the visited subtree is expanded according to the length to be traversed and RPs are identified on the subtree (5th step), then an approximate EST oracle is called to connect the current RPs, finally the additional length to be traversed in the next recursion is set to be half of the difference between L and the EST tour length (computed by the EST oracle) in the current recursion (7th step). The algorithm terminates if this difference becomes no larger than a threshold σ. The following proposition confirms the performance of RD–VT.

Proposition 10 *Let α be the best known approximation ratio for the GST problem and $\beta = L/\left(\sum_{(i,j)\in\mathcal{E}^*_N} d(i, j)\right)$, where $\mathcal{T}^*_N = \left(\mathcal{N}, \mathcal{E}^*_N\right)$ is the minimum GST of $\mathcal{N}$, the approximation ratio of the RD-VT algorithm is no greater than $\dfrac{\alpha - \beta/2}{1 - \beta}$.*

Proof For simplicity, we represent the total edge length of a tree $\mathcal{T}(\mathcal{V}, \mathcal{E})$ by $c(\mathcal{T}) = \sum_{(i,j)\in\mathcal{E}} d(i, j)$. Suppose the optimal set of RPs is $\mathcal{R}^*$ and its minimum GST is $\mathcal{T}^*_R$,

Algorithm 3 RD–VT

```
Input: Node set $\mathcal{N}$, tour length bound $L$, and threshold $\sigma$
1: Find an approximate GST of $\mathcal{N}$: $\mathcal{T}_N = (\mathcal{V}_N, \mathcal{E}_N)$, where $\mathcal{V}_N \supseteq \mathcal{N}$;
2: Initialize the tour length $\Gamma = L/2$ and a starting point $\Theta \in \mathcal{V}_N$;
3: repeat
4:     Traverse $\mathcal{T}_N$ in depth-first manner from $\Theta$ until the length visited is $\Gamma$, denote the subtree
       traveled as $\mathcal{T}_R = (\mathcal{V}_R, \mathcal{E}_R)$;
5:     Let $\mathcal{R} = \{r_i | r_i$ is the first intersection between $\mathcal{T}_R$ and the path from the $i$th node in $\mathcal{N}$ to
       $\Theta$ on $\mathcal{T}_N\}$;
6:     Find the ETS tour $\mathcal{U}$ that goes through $\mathcal{R}$ and denote its length by $|\mathcal{U}|$;
7:     Update trial step: $\Gamma = \Gamma + \Delta$, where $\Delta = (L - |\mathcal{U}|)/2$;
8: until $\Delta \le \sigma$
9: return a set $\mathcal{R}$ of RPs and a set of trees $\{\mathcal{T}_k(\mathcal{V}_k, \mathcal{E}_k)\} = \mathcal{T}_N \setminus \mathcal{T}_R$
```

and the optimal set of routing trees is $\left\{\mathcal{T}_k^*\right\}$, we have $\sum_k c\left(\mathcal{T}_k^*\right) + c\left(\mathcal{T}_R^*\right) \ge c\left(\mathcal{T}_N^*\right)$, as the union of $\left\{\mathcal{T}_k^*\right\}$ and $\mathcal{T}_R^*$ is a GST. As we have discussed, GST gives a lower bound of EST and the tour length of EST is bounded above by L, meaning $c\left(\mathcal{T}_R^*\right) \le L$, thus $\beta = L/c\left(\mathcal{T}_N^*\right) \ge L/\left(\sum_k c\left(\mathcal{T}_k^*\right) + c\left(\mathcal{T}_R^*\right)\right) \ge L/\left(\sum_k c\left(\mathcal{T}_k^*\right) + L\right)$, which leads to $L \le \dfrac{\beta}{1-\beta} \sum_k c\left(\mathcal{T}_k^*\right)$. Therefore we have

$$\begin{aligned}
\sum_k c\left(\mathcal{T}_k^*\right) &\ge c\left(\mathcal{T}_N^*\right) - c\left(\mathcal{T}_R^*\right) \\
&\ge \frac{c(\mathcal{T}_N)}{\alpha} - L \\
&= \frac{c(\mathcal{T}_N) - c(\mathcal{T}_R)}{\alpha} + \frac{c(\mathcal{T}_R)}{\alpha} - L \\
&\ge \frac{\sum_k c(\mathcal{T}_k)}{\alpha} + \frac{L/2}{\alpha} - L \\
&\ge \frac{\sum_k c(\mathcal{T}_k)}{\alpha} + \frac{1-2\alpha}{2\alpha}\frac{\beta}{1-\beta}\sum_k c\left(\mathcal{T}_k^*\right)
\end{aligned}$$

The approximation ratio is hence $\dfrac{\sum_k c(\mathcal{T}_k)}{\sum_k c\left(\mathcal{T}_k^*\right)} \le \alpha + \dfrac{\beta(2\alpha - 1)}{2(1-\beta)} = \dfrac{\alpha - \beta/2}{1-\beta}$. Q.E.D.

For $\beta \le 0.56$ and $\alpha = \dfrac{2}{\sqrt{3}}$, the ratio is smaller than 2. Since β is usually small (otherwise the mobile sink may almost visit every source node), the performance of RD–VT is pretty satisfactory. In particular, if $\mathcal{N}$ includes the whole WSN, the minimum GST is actually a MINIMUM SPANNING TREE (MST) and hence $\alpha = 1$ (MST can be perfectly solved efficiently), in which case the performance of RD–VT is further improved.

18.5.3 Summary

In this section, we focus on exploiting entity mobility in the fast mobility regime. We first formulate the mobile relay problem based on the optimization framework described in Sect. 18.2. As the resulting problem is very hard and there is no known PTAS for it, we discuss instead a variation appearing in the literature. This variation admits a PTAS with satisfactory performance at a cost of several simplifying assumptions. It is still an open question if we can handle the problem efficiently in more general settings where some of these simplifying assumptions may not hold.

18.6 Conclusion

No matter how advanced the stage we are in designing WSNs, network lifetime and energy efficiency will always be recurring issues pertaining to WSNs. To fully utilize the limited energy reserve of sensor nodes, we have to really "think out of the box" and explore new approaches. In our opinion, actively exploiting entity mobility in WSNs is such an approach, and an optimization framework is a powerful tool to guide the designs using this approach. By providing an in-depth description of the construction and application of such a general optimization framework as well as the engineering insights we can acquire from it, we are hoping to stimulate the invention of new design methodologies for WSNs.

References

1. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communication Mag*, 40(8):104–112, 2002.
2. V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3): 544–562, 2004.
3. M.A. Batalin, M. Rahimi, Y. Yu, D. Liu, A. Kansal, G.S. Sukhatme, W.J. Kaiser, M.Hansen, G.J. Pottie, M.Srivastava, and D. Estrin. Call and Response: Experiments in sampling the Environment. In: *Proceedings of the 2nd ACM SenSys*, Baltimore, Maryland, USA, 2004.
4. A. Bogdanov, E. Maneva, and S. Riesenfeld. Power-aware base station positioning for sensor Networks. In: *Proceedings of the 23rd IEEE INFOCOM*, Hong Kong, China, 2004.
5. A. Chakrabarti, A. Sabharwal, and B. Aazhang. Using predictable observer mobility for power efficient design of sensor networks. In: *Proceedings of the 2nd IEEE IPSN*, Palo Alto, California, USA, 2003.
6. J.-H. Chang and L. Tassiulas. Energy conserving routing in wireless Ad-hoc networks. In *Proceedings of the 19th IEEE INFOCOM*, Tel Aviv, Israel, 2000.
7. I. Chatzigiannakis, A. Kinalis, S. Nikoletseas, and J. Rolim. Fast and energy efficient sensor data collection by multiple mobile sinks. In: *Proceedings of the 5th ACM MobiWAC*, Chania, Crete Island, Greece, 2007.
8. D.-Z. Du, Y. Zhang, and Q. Feng. On better heuristic for euclidian steiner minimum trees. In *Proceedings of the 32nd IEEE FOCS*, San Juan, Puerto Rico, 1991.
9. R.W. Floyd. Algorithm 97. Shortest path. *Communications ACM*, 5(6):345, 1962.

10. L.R. Ford and D.R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, 1962.
11. S.R. Gandham, M. Dawande, R. Prakash, and S. Venkatesan. Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In: *Proceedings of IEEE Globecom*, San Francisco, California, USA, 2003.
12. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, NY, 1979.
13. N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In: *Proceedings of the 38th IEEE FOCS*, Miami Beach, Florida, USA, 1997.
14. M. Grossglauser and D. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Transactions on Networking*, 10(4):477–486, 2002.
15. Y. Gu, D. Bozda, E. Ekici, F. Ozguner, and C. Lee. Partitioning-based mobile element scheduling in wireless sensor networks. In: *Proceedings of the 2nd IEEE SECON*, Santa Clara, California, USA, 2005.
16. H.S. Kim, T.F. Abdelzaher, and W.H. Kwon. Minimum energy asynchronous dissemination to mobile sinks in wireless sensor networks. In: *Proceedings of the 1st ACM SenSys*, Los Angeles, California, USA, 2003.
17. A. Kinalis and S. Nikoletseas. Adaptive redundancy for data propagation exploiting dynamic sensory mobility. In: *Proceedings of the 11th ACM MSWiM*, 2008. Also in Journal of Interconnection Networks (JOIN), Vancouver, British Columbia, Canada, 2010.
18. L. Li, J.Y. Halpern, P. Bahl, Y.-M. Wang, and R. Wattenhofer. Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks. In: *Proceedings of the 20th ACM PODC*, Newport, Rhode Island, USA, 2001.
19. N. Li and J. Hou. Topology control in heterogeneous wireless networks: Problems and Solutions. In: *Proceedings of the 23rd IEEE INFOCOM*, Hong Kong, China, 2004.
20. Q. Li, M. De Rosa, and D. Rus. Distributed algorithms for guiding navigation across a sensor network. In: *Proceedings of the 9th ACM MobiCom*, San Diego, California, USA, 2003.
21. J. Luo and J.-P. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In: *Proceedings of the 24th IEEE INFOCOM*, Miami, Florida, USA, 2005.
22. J. Luo and J.-P. Hubaux. Joint sink mobility and routing to increase the lifetime of wireless sensor networks: The case of constrained mobility. *IEEE/ACM Transactions on Networking*, 18(3), June 2010.
23. J. Luo, J. Panchard, M. Piórkowski, M. Grossglauser, and J.-P. Hubaux. MobiRoute: Routing towards a mobile sink for improving lifetime in sensor networks. In: *Proceedings of the 2nd IEEE/ACM DCOSS*, San Francisco, California, USA, 2006.
24. M. Ma and Y. Yang. SenCar: An energy-efficient data gathering mechanism for large-scale multihop sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(10):1478–1488, 2007.
25. S. Madden, M.J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A tiny aggregation service for Ad-Hoc sensor networks. In: *Proceedings of the 5th USENIX OSDI*, Boston, MA, USA, 2002.
26. G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, NY, 1988.
27. V. Rodoplu and T. H. Meng. Minimum energy mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1333–1344, 1999.
28. R.C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling a three-tier architecture for sparse sensor networks. In: *Proceedings of the 1st IEEE SNPA*, Anchorage, Alska, USA, 2003.
29. F. Shahrokhi and D.W. Matula. The maximum concurrent flow problem. *Journal of ACM*, 37(2):318–334, 1990.
30. Y. Shi and Y.T. Hou. Theoretical results on base station movement problem for sensor Network. In *Proceedings of the 27th IEEE INFOCOM*, 2008.
31. Y. Shi, Y.T. Hou, and A. Efrat. Algorithm design for a class of base station location problems in sensor networks. *Springer Wireless Networks*, 15(1):21–38, 2009.

32. A. Somasundara, A. Kansal, D.D. Jea, D. Estrin, and M.B. Srivastava. Controllably mobile infrastructure for low energy embedded networks. *IEEE Transactions on Mobile Computing*, 5(8):958–973, 2006.
33. A.A. Somasundara, A. Ramamoorthy, and M.B. Srivastava. Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In: *Proceedings of the 25th IEEE RTSS*, Lisbon, Portugal, 2004.
34. G. Wang, G. Cao, and T. La Porta. Movement-assisted sensor deployment. *IEEE Transactions on Mobile Computing*, 5(6):640–652, 2006.
35. W. Wang, V. Srinivasan, and K.-C. Chua. Using mobile relays to prolong the lifetime of wireless sensor networks. In: *Proceedings of the 11th ACM MobiCom*, Cologne, Germany, 2005.
36. G. Xing, T. Wang, W. Jia, and M. Li. Rendezvous design algorithms for wireless sensor networks with a mobile base station. In: *Proceedings of the 9th ACM MobiHoc*, 2008.
37. F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A Two-Tier data dissemination model for large-scale wireless sensor networks. In: *Proceedings of the 8th ACM MobiCom*, Atlanta, Georgia, USA, 2002.

Part VI
Mobility Management

Chapter 19
Information Spreading in Dynamic Networks: An Analytical Approach

Andrea Clementi and Francesco Pasquale

Abstract Sensor networks are one of the most relevant concrete examples of dynamic networks. Their dynamic behavior is mainly due to the presence of node/link faults and node mobility. The aim of this chapter is to survey a new approach to study such dynamic networks, recently introduced in [15–19]. The major novelty of this approach relies on two basic issues.
1. The dynamic network is modeled as an *evolving graph* whose topology changes at every time according to some law/adversary. Both worst-case adversarial scenarios and graphs that evolve randomly are deeply studied.
2. This new approach provides a general framework where it is possible to determine the speed of *information spreading* from an *analytical* point of view.
Does the dynamic unknown behavior of sensor networks always slow down the speed of information spreading? What is the real impact of this dynamic behavior on the completion time of some basic communication protocols? Can unknown random node mobility be exploited to asymptotically speedup information spreading?
This new general approach provides some clean mathematical answers to the above fundamental questions.

19.1 Introduction

Sensor networks have an intrinsic *dynamic* behavior: their topology uses to *evolve over time*. Topology changes are caused by several factors, two of the most critical ones are *faults* and *node mobility*.

This chapter is not a survey of the most relevant models and results for dynamic sensor networks. Rather, it aims to describe a new mathematical approach to analyze the impact of such dynamic behavior on the *speed of information spreading*. Dynamic sensor networks have been previously investigated in many different

A. Clementi (✉)
Dipartimento di Matematica, Università di Roma "Tor Vergata", Via della Ricerca Scientifica, I-00133 Roma, Italy
e-mail: clementi@mat.uniroma2.it

contexts and from several points of view [23, 31, 34, 35, 38]. We here provide a general framework where a simple communication primitive can be asymptotically analyzed in order to better understand the main structural features of such dynamic systems and eventually exploit them in the protocol design.

As a first step, we thus have to define abstract models of concrete scenarios. The foundational nature of this approach introduces a crucial trade-off: the more a model is *realistic*, the more it is hard to analyze. Finding the right balance between these two poles is one of the hardest part of the work.

We model a dynamic network as an *evolving graph* [6, 8, 26]. In general, an evolving graph is a sequence of graphs (i.e., the *snapshots*) with the same set of nodes. Inspired by the two major sources of topology dynamicity mentioned above, we will consider three major classes of evolving graphs. The first two classes of evolving graphs are both defined by a random Markovian process: *Edge-Markovian Evolving graphs* and *Geometric-Markovian Evolving Graphs*. In the first class, the existence of every edge stochastically depends on its existence at the previous time step. As for the second Markovian class, the topology changes according to the current positions of the nodes and to their transmission range: every node performs a sort of random walk over a two-dimensional square of bounded size. Finally, we consider *Adversarial networks*: this is a worst-case scenario where link changes are managed by an adversary that tries to slow down communication protocols.

We will focus on the basic communication task called *broadcast*: one distinguished node of the network (the *source* node) aims to send a message to all the other nodes of the network. This chapter provides a revised version of the results presented in [15–19].

19.1.1 Warm-Up and Road Map

Let us consider the simplest broadcast protocol called *flooding* (or flooding mechanism). A node is informed if it is the source or if it has received the source message from one of its (informed) neighbors. Then, in the flooding protocol, every informed node sends the source message to all of its neighbors, at every time step.

The speed of information spreading can be then formalized by means of the following question: How long does it take to get all the nodes informed?

Given an evolving graph and a source node, the *completion time* of the flooding process is the first time step in which all the nodes are informed. The *flooding time* of an evolving graph is the maximum completion time over all possible source choices. The flooding process in a *static* graph looks exactly like a breadth-first search procedure. So, in a static graph, the flooding time equals the diameter of the graph; in particular flooding time is finite if and only if the graph is connected. Can we say something like that for evolving graphs? Is it true that, for example, if every snapshot of an evolving graph $\mathcal{G}$ has *small* diameter then the flooding time of $\mathcal{G}$ is *small*? In order to investigate such kind of issues in a clean way, let us make some thought experiments and introduce the concept of *adversarial* evolving graph.

19.1.1.1 Adversarial Evolving Graphs

How can a graph evolve in a *worst-case* way with respect to a communication task? Assume to have a set of nodes V and suppose there is an *adversary* that, at every time step t, can choose the set of edges E_t, thus yielding an evolving graph $\mathcal{G} = \{G_t : t \in \mathbb{N}\}$ where $G_t = (V, E_t)$ is the *snapshot* at time step t. Fix some constraints for the adversary, for example, all the snapshots must be connected and must have *small* diameter. For how long can the adversary slow down the flooding process? And, what is the flooding time of this adversarial evolving graph? On the one hand, it is clear that if all the snapshots are connected, then at every time step there is at least one new informed node. So the flooding time is upper bounded by the number of nodes. On the other hand, we may ask if it is possible to define an adversarial strategy such that at every time step the graph has *small* diameter, but the flooding time is still *big*. The answer is yes and finding an example is left as an easy exercise to the reader.

We say that an adversary is *meaningful* if the set of constraints makes the communication task *feasible* in the resulting adversarial evolving graph. For example, an adversary that makes all the snapshots connected is meaningful. We may ask whether this requirement is *minimal* or it can be relaxed. In other words, do we really need connectivity in order to complete the flooding process? No, we do not. Consider the following *weaker* connectivity constraint: at every time step, there must exist at least one link connecting a non-informed node with an informed one. It is easy to see that if the adversary satisfies this requirement, even if all the snapshots of the resulting evolving graph are fully disconnected, the flooding time is still upper bounded by the number of nodes. We say that a meaningful adversary with such a *weak connectivity* constraint is a *worst-case* adversary.

The above remarks show that concepts like *connectivity* and *diameter*, issues strongly related to communication tasks in static graphs, should be revised in evolving graphs. In a static graph, the spread of information is faster in graphs with small diameter, whereas if the graph is not connected, then there is no way to communicate. We have just pointed out that both those observations are false in evolving graphs. In Sect. 19.4, we will present a simple randomized broadcasting protocol and its full analysis for the worst-case adversarial model. The worst-case adversarial model and, thus, the obtained results consider the *interference* phenomenon too [4].

19.1.1.2 Random Evolving Graphs: Faults and Mobility

Graphs that evolve randomly are not a new subject in the networking area. For instance, a lot of effort has been devoted to define random *mobility* models that well approximate realistic mobile sensor networks [9, 30]. Typically, the network-evolution model is suitably chosen according to the particular application we are studying. Most of these models are very useful to analyze concrete communication problems and perform accurate and meaningful simulations; on the other hand, they are often too complicated to allow a general and deep mathematical analysis of basic communication tasks.

In this chapter, we instead adopt a different approach: starting from a random graph model that is well established in the static case, we try to extend it to the dynamic case inspired by *faulty* sensor networks and *mobile* sensor networks.
One of the most elegant and widely studied model of random graphs is the Erdös–Rényi model $G_{n,p}$ [25]. Informally speaking, it can be described as follows: given a set of n nodes, every edge between two nodes exists with probability p, independently from the other edges. As p grows from 0 to 1, the random graph becomes denser and denser, and the expected number of edges in the graph is p times the total number of possible edges. The properties of random graphs $G_{n,p}$ have been extensively studied [7, 42]. In order to analyze communication problems on random evolving graphs, our first purpose is to define a natural evolving version of the $G_{n,p}$ model motivated by the presence of random link failures.

In order to define a *reasonable* evolving graph, we observe that the state of the links at a given time step must somehow depend on the state of the links at the previous time steps. The simplest way to introduce this time dependence in a random process is to consider *Markovian* dependence: the random graph at time step t (only) depends on the random graph at time step $t - 1$. Let us take this property and try to "mix" it with the Erdös–Rényi model. Since in a $G_{n,p}$ the links are independent, it seems natural to consider the following model of random evolving graph: start with an *arbitrary* initial graph and, at every time step, if an edge exists then it will *die* at the next time step with probability q, while if a link does not exist, then it will *appear* in the next time step with probability p. We call this model *edge-Markovian evolving graph* (edge-MEG, in short).

In Sect. 19.2, we present an almost-tight analysis on the flooding time of edge-MEGs. The analysis provides a deep insight of the roles played by the crucial parameters of this evolving model. In particular, we will see that the role of the *death rate* q is almost negligible w.r.t. that of the *birth rate* p; moreover, a key ingredient in the information spreading process is the *dynamic expansion* property of the edge-MEG: a dynamic version of the classic concept of *node expansion* of static graphs. The above analysis concerns the flooding time of edge-MEGs with respect to the *worst-case* initial graph. One natural question is the following: What happens if the initial graph is "already" random according to the stationary distribution of the Markov chain that defines the process? In this case, all the snapshots of the evolving graph have the same marginal distribution. A random process with this property is called *stationary* [1]. In experimental papers, this situation is often referred to as *perfect simulation* [33].

In Sect. 19.3 we thus study stationary edge-MEGs and realize that our approach can be used to give upper bounds on the flooding time of a very general class of models recently introduced in [3] and called *Markovian evolving graphs*. Roughly speaking, a Markovian evolving graph (MEG, in short) is a Markov chain with a finite set of graphs as state space. Clearly, an edge-MEG is a very restricted case of MEG.

Let us now consider the following model called *geometric-MEG*. Take a set of n nodes over a two-dimensional grid, each of them performing a random walk. Consider the random evolving graph where there is an edge between two nodes if their

Euclidean distance is less than some fixed parameter r (i.e., the transmission range). Clearly, a geometric-MEG is a special case of MEG as well. Moreover it is a sort of prototype for a lot of random *mobility* models of sensor networks where the *physical* random node movements over some metric space determine the evolution of the *overlay* random graph: the *random walk* model [22, 28] and the *Walkers* model [21] are important examples of MEGs. Previous works on any basic communication task on such mobility models do not provide any relevant analytical result.

In Sect. 19.3.4, we apply our general method for MEG to analyze geometric-MEG. We then obtain almost-tight bounds for the flooding time of the latter when the stationary graph is (with high probability) connected.
What happens when this stationary graph is instead (with high probability) sparse and disconnected? This question captures the important phenomenon of *opportunistic networks* [27, 36] where information spreading is significantly helped by high node mobility rather than radio transmissions of large ranges. In this case, we cannot exploit any expansion property of the snapshots. In Sect. 19.3.5, we address this question by studying geometric-MEG where node transmission range is under the connectivity threshold while node speed is relatively high. We derive almost tight bounds on the flooding time that provide a mathematical evidence of the fact that, in some sensor scenarios, random node mobility *significantly helps* information spreading.

19.2 Edge-Markovian Evolving Graphs

An *evolving graph* $\mathcal{G}$ is a sequence of graphs with the same set of nodes V,

$$\mathcal{G} = \{G_t \ : \ t \in \mathbb{N}\} \quad \text{where } G_t = (V, E_t) \text{ and } E_t \subseteq \binom{V}{2}$$

A *random evolving graph* is a probability distribution over a family of evolving graphs. We introduce the following model of random evolving graph, the *edge-Markovian evolving graph*, that is a natural evolving version of the Erdös–Rényi random graph model $G_{n,p}$. Starting from an *arbitrary* initial graph $G_0 = ([n], E_0)$ with n nodes, at every time step every edge changes its state (existing or not) according to a two-state Markovian process with probabilities $p(n)$ and $q(n)$ independently of the other edges. If an edge exists at time t then at time $t+1$, it dies with probability $q(n)$. If instead the edge does not exist at time t then it will come into existence at time $t+1$ with probability $p(n)$. For brevity's sake, edge birth rate $p(n)$ and edge death rate $q(n)$ will be simply denoted as p and q, respectively.[1] Formally, we have the following definition.

[1] Hence, any inequality $p \leqslant (\geqslant) b(n)$ means that $p(n)$ is eventually not larger (not smaller) than $b(n)$. The same holds for $q = q(n)$.

Definition 1 (Edge-Markovian Evolving Graph) Given a positive integer $n \in \mathbb{N}$, two probabilities $0 \leqslant p, q \leqslant 1$, and a set $E_0 \subseteq \binom{[n]}{2}$, the *edge-Markovian evolving graph* $\mathcal{G}(n, p, q, E_0)$ is the random evolving graph

$$\mathcal{G}(n, p, q, E_0) = \{G_t = ([n], E_t) \ : \ t \in \mathbb{N}\}$$

where

$$E_t = \left\{ e \in \binom{[n]}{2} \ : \ X_t(e) = 1 \right\}$$

and for every $e \in \binom{[n]}{2}$, $\{X_t(e) \ : \ t \in \mathbb{N}\}$ is a Markov chain with transition matrix

$$M = \left(\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1-p & p \\ 1 & q & 1-q \end{array} \right)$$

and with initial condition $X_0(e) = 1$ if and only if $e \in E_0$. Markov chains $\left\{ X_t(e) \ : \ e \in \binom{[n]}{2} \right\}$ are independent.

Observe that setting $q = 1 - p$ yields the random evolving graph where links, at every time, are chosen independently at random, that is, a sequence of independent Erdös–Rényi $G_{n,p}$.

The flooding process. In order to evaluate the speed of information spreading in evolving graphs, we use a simple procedure usually called *flooding*. Start with one single *informed* node and, during the *evolution* of the graph, when a non-informed node *gets in touch* with an informed one, it *collects* the information. More formally we give the following definition.

Definition 2 (Flooding Process) Let $\mathcal{G}$ be an evolving graph and let $s \in [n]$ be a node. The flooding process in $\mathcal{G}$ with source s is the sequence $\{I_t \ : \ t \in \mathbb{N}\}$ of sets of nodes defined recursively as follows:

- $I_0 = \{s\}$ (at the beginning only the source node is *informed*);
- $I_{t+1} = I_t \cup N_t(I_t)$ (the neighbors of informed nodes become informed)

where $N_t(I_t)$ is the neighborhood of I_t in graph G_t,

$$N_t(I_t) = \{v \in [n] \setminus I_t \ : \ \{u, v\} \in E \text{ for some } u \in I_t\}$$

Given an evolving graph $\mathcal{G}$ and a source node $s \in [n]$, the *completion time* $T(\mathcal{G}, s)$ of the flooding process is the first time step in which all nodes of the network are informed. The *flooding time* $T(\mathcal{G})$ is the maximum completion time over all possible choices of source $s \in [n]$.

In this section we study the flooding time of edge-Markovian evolving graphs.

19.2.1 The Upper Bound

The flooding time of an evolving graph is strictly related to the *dynamic expansion properties* of the evolving graph (we will exploit this relation in the next section in a deeper way). Let us consider an edge-Markovian evolving graph $\mathcal{G} \sim \mathcal{G}(n, p, q, E_0)$. Let $I \subseteq [n]$ be a subset of nodes and let $t_0, t \in \mathbb{N}$ be two integers; we define $H_{t_0,t}(I)$ as the set of nodes (not in I) that have been connected to I in at least one time step $i \in \{t_0 + 1, \dots, t_0 + t\}$, i.e.,

$$H_{t_0,t}(I) = \left\{ v \in [n] \setminus I \;:\; \begin{array}{l} \{u, v\} \in E_i \\ \text{for some } u \in I \text{ and for} \\ \text{some } i = t_0 + 1, \dots, t_0 + t \end{array} \right\} \tag{19.1}$$

The size of set $H_{t_0,t}(I)$ somewhat evaluates the *dynamic expansion* property of the evolving graph. A good lower bound on such parameter is used in the proof of the next lemma. If I is the set of informed nodes at time t_0 then all nodes in $H_{t_0,t}(I)$ will be informed within time $t_0 + t$. Indeed, from Definition 2 and from definition of set $H_{t_0,t}$ in (19.1), it holds that

1. For any $t \in \mathbb{N}$, $I_{t-1} \subseteq I_t$;
2. For any $t_0, t \in \mathbb{N}$, $I_{t_0} \cup H_{t_0,t}(I_{t_0}) \subseteq I_{t_0+t}$.

In order to evaluate the size of I_{t_0+t}, we will always use property 2 above.

Lemma 1 *After $t = O\left(\dfrac{\log n}{\log(1 + np)}\right)$ time steps, the number m_t of informed node is at least βn w.h.p., where β is a positive constant.*

Idea of the Proof Consider the stochastic process $\{m_t \;:\; t \in \mathbb{N}\}$, where m_t is the random variable counting the number of informed nodes at time step t. We want to evaluate how large t must be in order to have that m_t is at least a constant fraction of all the nodes w.h.p.

Assume we could prove that for every fixed t, the following relation holds w.h.p

$$m_{t+1} \geqslant (1 + np) m_t \tag{19.2}$$

This inequality could be obtained by providing a lower bound on the size of $H_{t,1}(I_t)$. Informally speaking, Eq. (19.2) says that the number m_{t+1} of informed nodes at time step $t + 1$ is at least the number m_t of informed nodes at the previous time step, plus some multiple (or fraction, depending on how large np is) of m_t itself. Then, by using union bound and some technical effort, we could prove that such a relation holds w.h.p. for every t in a sufficiently large range, and iterating we would have w.h.p.

$$m_t \geqslant (1 + np)^t m_0$$

Since $m_0 = 1$, it would easily follow that in order to have $m_t \geqslant \beta n$, it is sufficient to have $t \geqslant \dfrac{\log \beta n}{\log(1+np)}$, that is, exactly what we want to prove.
Unfortunately, recurrence 19.2 does not hold for every t, but only for some ranges of m_t and p. However, if recurrence 19.2 is false, it happens for one of the following two reasons: (1) m_t is *too large*, so that $np\, m_t$ is greater than n; (2) m_t is *too small*, so that even if recurrence 19.2 holds *in expectation* (i.e., we have $\mathbf{E}\left[m_{t+1}\right] \geqslant (1+np)\mathbf{E}\left[m_t\right]$), we cannot prove concentration results to achieve the high probability that we need.
The first case is easy to handle because, when m_t is sufficiently large, we can prove that in the next time step the number of informed nodes will be a constant fraction of all nodes, with high probability. As for the second case, waiting a suitable amount of time τ, i.e., by evaluating the size of $H_{t,\tau}(I_t)$ instead of $H_{t,1}(I_t)$, the number of new informed nodes $m_{t+\tau}$ is significantly bigger than m_t. □

Once we have a constant fraction of informed nodes, it is easy to show that after further $O\left(\dfrac{\log n}{np}\right)$ time steps all nodes will be informed w.h.p.

Lemma 2 *Let* $0 < \beta < 1$ *be any constant and* $m \geqslant \beta n$ *be the number of informed nodes at time step* $t_0 \geqslant 0$. *Then, after* $O\left(\dfrac{\log n}{np}\right)$ *time steps, all nodes will be informed w.h.p.*

Idea of the Proof Let v be a non-informed node. The probability that node v will not be a neighbor of anyone of the βn informed nodes in anyone of the next t time steps is less than $(1-p)^{\beta n t}$. Hence, for $t \geqslant c \log n/(\beta n p)$, such probability becomes less than n^{-c}. The thesis follows by taking the union bound over all $(1-\beta)n$ non-informed nodes. □

By combining Lemmas 1 and 2 we have the following upper bound on the flooding time.

Theorem 1 *Let* $\mathcal{G} \sim \mathcal{G}(n, p, q, E_0)$ *be any edge-Markovian evolving graph. Then, the flooding time of* $\mathcal{G}$ *is* $O\left(\dfrac{\log n}{\log(1+np)}\right)$ *w.h.p.*

19.2.2 The Lower Bounds

In this section, we show that the upper bound in Theorem 1 is tight for a large range of the parameters. We will make use of the following natural "monotonicity" property: the flooding time decreases as birth rate p increases while it increases as death rate q increases.

Lemma 3 *Consider two edge-Markovian evolving graphs* $\mathcal{G} \sim \mathcal{G}(n, p, q, \emptyset)$ *and* $\mathcal{G}' \sim \mathcal{G}(n, p', q', \emptyset)$ *where*

$$p \leqslant p' ; \qquad q' \leqslant q ; \qquad p + q' \leqslant 1$$

Then, it holds that

$$\mathbf{P}\left(T(\mathcal{G}) \leqslant t\right) \leqslant \mathbf{P}\left(T(\mathcal{G}') \leqslant t\right) \qquad \text{for any} \quad t \in \mathbb{N}$$

Observe that the above result requires the somewhat "artificial" condition $p + q' \leqslant 1$. Indeed, its proof relies on a suitable application (stated in the next lemma) of the *coupling* technique of Markov chains that works only under the above condition. Proving the monotonicity property without this condition by using any other technique is an open problem which seems far from easy.

Lemma 4 *(Markov-coupling Lemma) Let* $\{Z_t = (X_t, Y_t) \,:\, t \in \mathbb{N}\}$ *be a Markov chain with state space* $\{(0,0),\ (0,1),\ (1,1)\}$ *and transition matrix*

$$M^Z = \left(\begin{array}{c|ccc} & (0,0) & (0,1) & (1,1) \\ \hline (0,0) & 1-p' & p'-p & p \\ (0,1) & q' & 1-p-q' & p \\ (1,1) & q' & q-q' & 1-q \end{array}\right) \tag{19.3}$$

where $p \leqslant p'$, $q \geqslant q'$, *and* $p + q' \leqslant 1$. *Then,* $\{X_t \,:\, t \in \mathbb{N}\}$ *and* $\{Y_t \,:\, t \in \mathbb{N}\}$ *are Markov chains with state space* $\{0, 1\}$ *and transition matrices, respectively,*

$$M^X = \left(\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1-p & p \\ 1 & q & 1-q \end{array}\right) \qquad\qquad M^Y = \left(\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1-p' & p' \\ 1 & q' & 1-q' \end{array}\right)$$

In order to obtain a general lower bound on the flooding time, as a first approximation we can prove a lower bound in the special case $q = 0$, and thanks to Lemma 3, it will be a lower bound for any $q \in [0, 1]$. At first this may seem quite a rough approximation; instead it turns out to be asymptotically tight for most of the cases, as we will see soon.

Notice that starting from the empty graph (i.e., $E_0 = \emptyset$), and if *edges do not die* (i.e., $q = 0$), the resulting edge-MEG $\mathcal{G}(n, p, 0, \emptyset)$ is just a sequence of Erdös–Rényi random graphs with edge probabilities depending on the time step t. Indeed, when $q = 0$ the Markov chain $\{X_t \,:\, t \in \mathbb{N}\}$ that defines the edge-Markovian evolving graph has transition matrix

$$\left(\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1-p & p \\ 1 & 0 & 1 \end{array}\right)$$

And given the initial condition $E_0 = \emptyset$, that yields $X_0 = 0$ with probability 1, with an easy calculation we obtain

$$\mathbf{P}(X_t = 1) = 1 - (1-p)^t \tag{19.4}$$

So if $\mathcal{G} = \{G_t : t \in \mathbb{N}\}$ is an edge-MEG $\mathcal{G} = \mathcal{G}(n, p, 0, \emptyset)$, then G_0 is the empty graph by definition, G_1 is a $G_{n,p}$ random graph, G_2 is a $G_{n,1-(1-p)^2}$, and in general G_t is a random graph G_{n,p_t} with $p_t = 1 - (1-p)^t$. It is worth remarking that random graphs $\{G_t : t \in \mathbb{N}\}$ are *not independent*, but we are interested here in their marginal distributions.
Moreover, when $q = 0$ we have another important property that will turn out to be useful: The graph is *densifying*, i.e., $E_t \subseteq E_{t+1}$. This means that for example, if G_t is not connected, then $G_{t'}$ is not connected for every $t' \leqslant t$, more formally

$$\mathbf{P}(G_{t'} \text{ connected } \mid G_t \text{ not connected}) = 0 \quad \text{if } t' \leqslant t$$

Lemma 5 *Let $\mathcal{G}$ be an edge-Markovian evolving graph with initial condition $E_0 = \emptyset$ and death rate $q = 0$, $\mathcal{G} \sim \mathcal{G}(n, p, 0, \emptyset)$, then the flooding time of $\mathcal{G}$ is $\Omega\left(\frac{\log n}{np}\right)$ w.h.p.*

Idea of the Proof Graph G_t is an Erdös–Rényi G_{n,p_t} with $p_t = 1-(1-p)^t \approx pt$. It is well-known that if $p_t \leqslant (1-\varepsilon)\log n/n$ random graph G_{n,p_t} is not connected w.h.p. Hence, in order to have flooding be completed at time t w.h.p., G_t must be connected w.h.p. and so t must be $\Omega(\log n/(np))$. □

Lemma 6 *Let $\mathcal{G}$ be an edge-Markovian evolving graph with initial condition $E_0 = \emptyset$ and death rate $q = 0$, $\mathcal{G} \sim \mathcal{G}(n, p, 0, \emptyset)$. If $p \geqslant \frac{\log n}{n}$ then the flooding time of $\mathcal{G}$ is $\Omega\left(\frac{\log n}{\log(np)}\right)$ w.h.p.*

Idea of the Proof Since $q = 0$ edges *do not die*, i.e., $E_t \subseteq E_{t+1}$ for every t. Since $E_0 = \emptyset$ it is easy to see that if the flooding process is completed at time t, then the diameter of G_t is at most $2t$. Random graph G_t is an Erdös–Rényi G_{n,p_t} with $p_t = 1-(1-p)^t \approx pt$ and it is well known [13] that the diameter of a connected random graph G_{n,p_t} is $\Theta\left(\frac{\log n}{\log(np_t)}\right)$. Hence if $\bar{t}$ is the flooding time, it must satisfy $\bar{t} = \Omega\left(\frac{\log n}{\log(np\bar{t})}\right)$.
By hypothesis $p \geqslant \frac{\log n}{n}$ thus $\frac{\log n}{\log(np\bar{t})} \leqslant \frac{\log n}{\log\log n}$. Since $\log(np\bar{t}) = \Theta(\log(np))$ whenever $\bar{t} = O(\text{poly }(np)) = O(\text{polylog } n)$ it follows that $\bar{t} = \Omega\left(\frac{\log n}{\log(np)}\right)$. □

We can now consider arbitrary death rate q. Indeed, from Lemma 3, Lemma 5, and Lemma 6, we obtain the two following lower bounds.

Theorem 2 *Let $\mathcal{G}$ be an edge-Markovian evolving graph with initial condition the empty graph $\mathcal{G} \sim \mathcal{G}(n, p, q, \emptyset)$. Then the flooding time of $\mathcal{G}$ is $\Omega\left(\frac{\log n}{np}\right)$ w.h.p.*

Observe that when $p = O(1/n)$, the previous lower bound matches the upper bound in Theorem 1 because $\log(1 + np) = \Theta(np)$ when $np = O(1)$.

Theorem 3 *Let $\mathcal{G}$ be an edge-Markovian evolving graph with initial condition the empty graph $\mathcal{G} \sim \mathcal{G}(n, p, q, \emptyset)$. If $p \geqslant \frac{\log n}{n}$, then the flooding time of $\mathcal{G}$ is $\Omega\left(\frac{\log n}{\log(np)}\right)$ w.h.p.*

Observe that this lower bound matches the upper bound in Theorem 1, because $\log(1 + np) = \Theta(\log(np))$ when $np = \omega(1)$.

Consequences of the bounds In Table 19.1 we summarize the results of this section, then we point out some observations arising from them.

(1) *Flooding can be exponentially faster than mixing.* As we will see in Sect. 19.3.2, the stationary distribution of any edge-MEG $\mathcal{G} = \mathcal{G}(n, p, q, E_0)$ with $0 < p, q < 1$ is an Erdös–Rényi random graph. We emphasize that our previous bounds on flooding time concern the (worst-case) *transient phase* of edge-MEG: in several cases, the flooding time can be exponentially shorter than the *mixing time* (see [14]).

(2) *Sparseness and disconnectivity do not prevent fast flooding.* Another important consequence of the upper bound in Theorem 1 is that starting from any initial distribution E_0, the flooding time can be logarithmic even for $p = o(\log n/n)$ (whatever is q), i.e., even below the *connectivity threshold* of the stationary random graph. In particular, the flooding time is at most logarithmic even when the expected "stationary" node degree is 1.

(3) *Flooding time may be independent from the initial graph.* When $p \geqslant 1/n^{\delta}$ for some constant $0 < \delta < 1$ and arbitrary q, the upper bound in Theorem 1 turns out to be constant regardless of the initial distribution E_0.

(4) *Tightness.* For $p \geqslant \log n/n$, the obtained upper bound and lower bound are *tight* in the standard "computational-complexity" sense: the upper bound holds

Table 19.1 Flooding time of edge-MEG $\mathcal{G}(n, p, q, E_0)$

$p < \frac{1}{n}$	$\frac{1+\varepsilon}{n} < p < \frac{\log n}{n}$	$p > \frac{\log n}{n}$
$\Theta\left(\frac{\log n}{np}\right)$	$O\left(\frac{\log n}{\log(np)}\right)$	$\Theta\left(\frac{\log n}{\log(np)}\right)$
	$\Omega\left(\frac{\log n}{np}\right)$	

for *any* initial graph while the lower bound is satisfied by *at least one* initial graph and they are asymptotically tight. We can state the same tightness for $p \leqslant 1/n$ since, in this case, $np = \Theta(\log(1 + np))$.

(5) *The role of the death rate.* In Sect. 19.2.1, we proved an upper bound (see Theorem 1) on the flooding time of edge-MEGs. Our upper bound does not depend on the death rate q. Despite this seeming roughness, in Sect. 19.2.2 we proved that the upper bound is tight whenever $p \notin \left[\frac{1}{n}, \frac{\log n}{n}\right]$. Indeed, Theorems 2 and 3 state that an initial condition (i.e., the empty graph $E_0 = \emptyset$) exists such that the flooding time is lower bounded by two q-independent formulas: such formulas, outside the above p-range, match the upper bound. On the other hand, when $p \in \left[1/n,\ (\log n)/n\right]$ there is a gap between our upper and lower bounds. It turns out that when p is in that small range, the role of the death rate q is no longer *asymptotically negligible*. We do not know what actual asymptotic formula for the flooding time holds in that range; however, it is possible to prove it *must* depend on the death rate q as well (see [17]).

19.3 Stationary Markovian Evolving Graphs

Markovian evolving graphs are a natural and very general class of models for evolving graphs introduced in [3]. In these models, the set of nodes is fixed and the edge set at time t stochastically depends on the edge set at time $t - 1$.

Definition 3 (Markovian evolving graph [3]) Let $\mathbf{G}$ be a family of graphs with the same node set $[n]$. A *Markovian evolving graph* $\mathcal{M} = \{G_t : t \in \mathbb{N}\}$ is a Markov chain with state space $\mathbf{G}$.
A *stationary* Markovian evolving graph is a Markovian evolving graph $\mathcal{M} = \{G_t : t \in \mathbb{N}\}$ such that G_0 is random with a stationary distribution of $\mathcal{M}$.

In this section we study the flooding time of *stationary* Markovian evolving graphs. We prove an upper bound on the flooding time of *any* stationary Markovian evolving graph. This upper bound is expressed in terms of the parametrized node-expansion properties satisfied by the stationary graphs.
We then show the tightness of this bound in two relevant and natural dynamic scenarios: *edge-Markovian evolving graphs* (in short, *edge-MEG*) and *geometric Markovian evolving graphs* (in short, *geometric-MEG*).

19.3.1 Flooding Time and Expansion Properties

The following definition concerns a sort of parametrized node expansion. This is a key ingredient in the analysis of flooding in Markovian evolving graphs to cope with the difficulties due to stochastic dependence.

Definition 4 (Expander) A graph $G = ([n], E)$ is a (h, k)*-expander* if, for every set of nodes $I \subseteq [n]$ with $|I| \leqslant h$, it holds that $|N(I)| \geqslant k|I|$.

The above definition naturally extends to random variables and their probability distributions.

Definition 5 (Expander II) Let X be a random variable with values in a family of graphs with the same node set $[n]$. Then X is a (h, k)-*expander* with probability p if

$$\mathbf{P}\left(X \text{ is a } (h,k)\text{-expander}\right) \geqslant p$$

In this case, we also say that the probability distribution of X *yields* an (h, k)-expander with probability p.

We are now able to provide the main result for general stationary Markovian evolving graphs.

Theorem 4 *Let $\mathcal{M} = \{G_t : t \in \mathbb{N}\}$ be a stationary Markovian evolving graph. Assume an increasing sequence $1 = h_0 \leqslant h_1 < \cdots < h_s = n/2$ and a decreasing sequence $k_1 \geqslant \cdots \geqslant k_s$ of positive real numbers exist such that, for every $i = 1, \ldots, s$, the stationary distribution of $\mathcal{M}$ yields an (h_i, k_i)-expander with probability $1 - \frac{1}{n^4}$. Then the flooding time of $\mathcal{M}$ is w.h.p.*

$$O\left(\sum_{i=1}^{s} \frac{\log(h_i/h_{i-1})}{\log(1+k_i)}\right)$$

Idea of the Proof Since by the theorem's hypothesis, the distribution of G_0 is the stationary distribution of the Markov chain $\mathcal{M}$, for every $t \in \mathbb{N}$, r.v. G_t has the same distribution of G_0.
Let us call $m_t = |I_t|$ the number of informed nodes at time step t; at the beginning we have $m_0 = 1$. Since every r.v. G_t is an (h_1, k_1)-expander w.h.p. then, as long as $m_t \leqslant h_1$, the recurrence

$$m_{t+1} \geqslant (1+k_1)m_t \tag{19.5}$$

holds w.h.p. Indeed, whatever the stochastic dependence were till time step t, the node-expansion property guarantees that $|N(I_t)|$ is at least $k_1 m_t$. The closed form of the above recurrence is $m_t \geqslant (1 + k_1)^t m_0$, hence

$$O\left(\frac{\log(h_1/m_0)}{\log(1+k_1)}\right)$$

time steps are enough to get $m_t \geqslant h_1$ w.h.p. However, the latter bound might be $o(1)$ and this requires some technical care to be treated (see the full proof in [18]).
Now, let t_1 be the smallest time step such that $m_{t_1} \geqslant h_1$. From that time step on, we cannot use recurrence (19.5) anymore, but as G_t is also an (h_2, k_2)-expander w.h.p., as long as $m_t \leqslant h_2$, the new recurrence $m_{t+1+t_1} \geqslant (1 + k_2)m_{t+t_1}$ holds w.h.p. By solving the recurrence, we obtain $m_{t+t_1} \geqslant (1 + k_2)^t m_{t_1} \geqslant (1 + k_2)^t h_1$. So the number of time steps required to reach h_2 informed nodes is w.h.p.

$$O\left(\frac{\log(h_2/h_1)}{\log(1+k_2)}\right)+O\left(\frac{\log h_1}{\log(1+k_1)}\right)$$

We apply this way of reasoning over all sequence of expansion parameters and we thus get that at least $n/2$ nodes will be informed within a number of time steps that is w.h.p. within the bound of the theorem.
Once there are $n/2$ informed nodes, a symmetric argument shows that the number of non-informed nodes decreases at the same rate. □

Definition 3 naturally extends in the following way. We will need this generalization to include geometric-MEG (see Sect. 19.3.4).

Definition 6 (Markovian Evolving Graph II) Let **G** be a family of graphs with the same node set $[n]$. A *Markovian evolving graph* $\mathcal{G} = \{G_t \ : \ t \in \mathbb{N}\}$ is a sequence of random variables with state space **G** and such that there exist both a Markov chain $\mathcal{X} = \{X_t \ : \ t \in \mathbb{N}\}$ and a function f so that $G_t = f(X_t)$.
A *stationary* Markovian evolving graph is a Markovian evolving graph $\mathcal{G} = \{G_t \ : \ t \in \mathbb{N}\}$ such that G_0 is random with a stationary distribution of $\mathcal{X}$ translated by f.

It is not hard to show that Theorem 4 easily extends to the above generalized definition of Markovian evolving graphs.

19.3.2 Stationary Edge-MEGs

We recall the model introduced in the previous chapter. A stationary edge-MEG $\mathcal{M}(n, p, q) = \{G_t \ : \ t \in \mathbb{N}\}$ is a Markov chain such that $G_t = ([n], E_t)$ with

$$E_t = \left\{ e \in \binom{[n]}{2} \ : \ X_t(e) = 1 \right\}$$

where $\{X_t(e) \ : \ e \in \binom{[n]}{2}\}$ are independent Markov chains with transition matrix

$$M = \left(\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1-p & p \\ 1 & q & 1-q \end{array}\right)$$

and for every $e \in \binom{V}{2}$, $X_0(e)$ is random according to the stationary distribution of M. Clearly, a stationary edge-MEG is a stationary Markovian evolving graph according to Definition 3. Observe that the stationary distribution of the Markov chains $\{X_t(e) \ : \ t \in \mathbb{N}\}$ is

$$\pi_e = \left(\frac{q}{p+q}, \frac{p}{p+q}\right)$$

Hence, the stationary distribution of $\mathcal{M}(n, p, q)$ is $G_{n,\hat{p}}$ (i.e., Erdös–Rényi distribution in which each possible edge occurs independently with probability $\hat{p}$) where here and in the sequel

$$\hat{p} = \frac{p}{p+q}$$

Theorem 5 *Let $\mathcal{M}(n, p, q)$ be a stationary edge-MEG such that $\hat{p} \geqslant c\frac{\log n}{n}$ for a sufficiently large constant c. Then the flooding time of $\mathcal{M}(n, p, q)$ is w.h.p.*

$$O\left(\frac{\log n}{\log(n\hat{p})} + \log\log(n\hat{p})\right)$$

Idea of the Proof The stationary distribution of an edge-MEG is an Erdös–Rényi random graph, whose expansion properties are well known and easy to calculate. The thesis follows by applying Theorem 4 with such expansion parameters. □

The next theorem gives a lower bound on the flooding time of stationary edge-MEGs.

Theorem 6 *Let $\mathcal{M}(n, p, q)$ be a stationary edge-MEG such that $\hat{p} \geqslant c\frac{\log n}{n}$ for a sufficiently large constant c. Then the flooding time of $\mathcal{M}(n, p, q)$ is w.h.p.*

$$\Omega\left(\frac{\log n}{\log(n\hat{p})}\right)$$

Idea of the Proof For a fixed time step t, the degree of any node in the network is $\Theta(n\hat{p})$ w.h.p. By taking the union bound we have that it holds w.h.p. for a sufficiently long sequence of time steps, say $\Theta(n)$ time steps. Hence, the number of new informed nodes at every time step is at most $\Theta(n\hat{p})$ times the current number of informed nodes w.h.p. □

Consequences of the bounds In Sect. 19.2, the *maximal* flooding time in edge-MEG has been studied with respect to *any* initial probability distribution. However, those results do not say whether flooding can be (significantly) faster in *stationary* edge-MEG. Interestingly enough, the *stationary* bound implies that whenever the birth rate p is $O(1/n^{1+\varepsilon})$ and the death rate q is $O(np/\log n)$, there is an *exponential* gap between the stationary case and the worst case. An exponential gap also holds whenever $p = O(\log n/n)$ and $q = O\left(p\sqrt{n}\right)$ (for instance, set $q = (\text{polylog}\, n)/n$).

19.3.3 Parsimonious Flooding in Stationary Edge-MEGs

The flooding process can be seen as a broadcasting protocol where, when a node gets the source message, the node forwards it for *all* the following time steps. In [5] the

authors introduce a variant of this protocol in which every node forwards the source message only for the next k time steps after the message reception: this variant is called the k-active flooding protocol.
Two main questions arise:

1. How large k must be in order to guarantee (w.h.p.) that the flooding process is completed?
2. When k is large enough to accomplish the task in (1), what is the completion time of the k-active flooding, compared to the *unrestricted* flooding?

Let $\mathcal{M}(n, p, q) = \{G_t \ : \ t \in \mathbb{N}\}$ be a stationary edge-MEG and let $s \in [n]$ be a source node. The *completion time* of the k-active flooding protocol is the first time step (if any) such that all nodes are informed

$$T_s^{(k)} = \infty\{t \geqslant 0 \ : \ I_t = [n]\}$$

The *reachability threshold* for the flooding protocol is the smallest integer k such that $T_s^{(k)} < \infty$ almost surely (a.s. in short)[2] for any $s \in [n]$.
Informally speaking, it turns out that (i) when $\hat{p}$ is large enough, namely greater than $\log n/n$, $k = 1$ suffices to complete flooding a.s.; (ii) when $1 < n\hat{p} < \log n/n$, the reachability threshold k is a decreasing function of the birth rate p; (iii) when $\hat{p} < 1/n$ all nodes must be active for the whole execution of the protocol in order to guarantee the flooding task a.s. Interestingly enough, whenever k is sufficiently large to guarantee the flooding task, the completion time of the k-active flooding protocol is the same as the completion time of the unrestricted flooding protocol up to a constant factor.
More formally, in the following theorem we summarize the results in [5].

Theorem 7 *Let $\mathcal{M}(n, p, q) = \{G_t \ : \ t \in \mathbb{N}\}$ be a stationary edge-MEG and let $\hat{p} = \dfrac{p}{p+q}$ be the stationary edge probability.*

1. *If $\hat{p} = \omega\left(\dfrac{\log n}{n}\right)$ then the reachability threshold is $k = 1$ and the 1-active flooding time is $O\left(\dfrac{\log n}{\log n\hat{p}}\right)$ a.s.;*
2. *If $1/n \leqslant \hat{p} \leqslant c\frac{\log n}{n}$ then*
    - *The reachability threshold is $\Theta\left(\dfrac{\log n}{np}\right)$ a.s.;*
    - *For any $k = \Omega\left(\dfrac{\log n}{np}\right)$ the k-active flooding time is $\Theta\left(\dfrac{\log n}{np} + \dfrac{\log n}{n\hat{p}}\right)$ a.s.;*

[2] *Almost surely* here means with probability that tends to 1 as n goes to infinity, without requiring the rate of convergence to be the inverse of a polynomial in n, as it is for *w.h.p.*

3. *If* $\hat{p} \leqslant c/n$ *then both the reachability threshold and the flooding time are* $\Theta\left(\frac{\log n}{np}\right)$ *a.s.*

Idea of the Proof The main step in the proof is a *reduction lemma* in which the authors show that the completion time of the k-active flooding process in stationary edge-MEG $\mathcal{M}(n, p, q)$ is equal to the diameter of a suitable weighted random graph $G^{(k)}_{n,\hat{p},p}$. The analysis of the reachability threshold and of the k-active flooding time is then obtained by analyzing the connectivity property and the diameter of this weighted random graph $G^{(k)}_{n,\hat{p},p}$. □

19.3.4 Stationary Geometric-MEGs

We introduce a model of evolving graphs that is a discrete version of the *random walk mobility* model for radio networks [9]. In the latter model, nodes (i.e., radio stations) move on a bounded region of the plane (typically a square region) and each node performs, independently from the others, a sort of Brownian motion. At any time there is an edge (i.e., a bidirectional connection link) between two nodes if they are at distance at most r (in the language of radio networks it represents the *transmission range*). In our model we discretize time and space. We choose to keep constant the density (i.e., the ratio between the number of nodes and the area) as the number n of nodes grows. The region in which nodes move is a square of side $\sqrt{n}$ and the density equals 1. We remark that this choice is only for the sake of convenience and all the results can be scaled to any density $\delta(n)$. The nodes can assume positions whose coordinates are integer multiple of a resolution coefficient $\varepsilon>0$. Formally, nodes move on the following set of points:

$$L_{n,\varepsilon} = \left\{(i\varepsilon, j\varepsilon) \mid i, j \in \mathbb{N} \wedge i, j \leqslant \frac{\sqrt{n}}{\varepsilon}\right\}$$

At any time step, a node can move to one of the positions of $L_{n,\varepsilon}$ within distance ρ from the previous position. The positive real number ρ is a fixed parameter that we call *move radius*. It can be interpreted as the maximum velocity of a node.[3] Formally, we introduce the *move graph* $M_{n,\rho,\varepsilon} = (L_{n,\varepsilon}, E_{n,\rho,\varepsilon})$, where

$$E_{n,\rho,\varepsilon} = \{(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in L_{n,\varepsilon} \;\; d(\mathbf{x}, \mathbf{y}) \leqslant \rho\}$$

and $d(\cdot, \cdot)$ is the Euclidean distance. A node in position $\mathbf{x}$, in one time step, can move in any position in $\Gamma(\mathbf{x})$, where $\Gamma(\mathbf{x}) = \{\mathbf{y} \mid (\mathbf{x}, \mathbf{y}) \in E_{n,\rho,\varepsilon}\}$. The nodes are identified by the first n positive integers $[n]$. The time evolution of the movement of a single node i is represented by a Markov chain $\{P_{i,t} \; ; \; t \in \mathbb{N}\}$ where $P_{i,t}$ are random variables whose state space is $L_{n,\varepsilon}$ and

[3] Indeed, a node can run through a distance of at most ρ in a unit of time.

$$\mathbf{P}\left(P_{i,t+1} = \mathbf{x}\right) = \begin{cases} \frac{1}{|\Gamma(P_{i,t})|} & \text{if } \mathbf{x} \in \Gamma(P_{i,t}) \\ 0 & \text{otherwise} \end{cases}$$

In other words, $P_{i,t}$ is the position of node i at time t. Thus, the time evolution of the movements of all the nodes is represented by a Markov chain $\mathcal{P}(n, \rho, \varepsilon) = \{P_t : t \in \mathbb{N}\}$ whose state space is $L_{n,\varepsilon} \times L_{n,\varepsilon} \times \cdots \times L_{n,\varepsilon}$ (n times) and

$$P_t = (P_{1,t}, P_{2,t}, \ldots, P_{n,t})$$

Let us fix a *transmission radius* $r>0$. A *geometric-MEG* is a sequence of random variables $\mathcal{G}(n, \rho, r, \varepsilon) = \{G_t : t \in \mathbb{N}\}$ such that $G_t = ([n], E_t)$ with

$$E_t = \{(i, j) \mid d(P_{i,t}, P_{j,t}) \leqslant r\}$$

Clearly, a geometric-MEG is a Markovian evolving graph according to Definition 6. As for the stationary case, we observe that the stationary distribution π_i of Markov chain $\{P_{i,t} \ ; \ t \in \mathbb{N}\}$ is (see for example [1])

$$\pi_i(\mathbf{x}) = \frac{|\Gamma(\mathbf{x})|}{\sum_{\mathbf{y} \in L_{n,\varepsilon}} |\Gamma(\mathbf{y})|}$$

Moreover, the stationary distribution of $\mathcal{P}(n, \rho, \varepsilon)$ is the product of the independent distributions π_i for all $i \in [n]$. We say that a geometric-MEG $\mathcal{G}(n, \rho, r, \varepsilon) = \{G_t : t \in \mathbb{N}\}$ is a *stationary* geometric-MEG if the underlying P_0 is random with the stationary distribution of the Markov chain $\mathcal{P}(n, \rho, \varepsilon) = \{P_t : t \in \mathbb{N}\}$. Notice that if $\mathcal{G}(n, \rho, r, \varepsilon) = \{G_t : t \in \mathbb{N}\}$ is a stationary geometric-MEG then all random variables G_t have the same probability distribution that we call *stationary distribution* of $\mathcal{G}(n, \rho, r, \varepsilon)$.

Theorem 8 *Let $\mathcal{G}(n, \rho, r, \varepsilon)$ be a stationary geometric-MEG. If $\varepsilon \leqslant 1$ and $c\sqrt{\log n} \leqslant r \leqslant \sqrt{n}$ for a sufficiently large constant c, then the flooding time of $\mathcal{G}(n, \rho, r, \varepsilon)$ is w.h.p.*

$$O\left(\frac{\sqrt{n}}{r} + \log\log r\right)$$

Idea of the Proof The stationary distribution of a geometric-MEG is *approximately* (up to a small *border effect*) a geometric random graph in which points are thrown uniformly and independently at random in the square. The expansion parameters of the geometric random graph (and of its approximate version) are well known and easy to calculate. The thesis follows by applying Theorem 4 with such expansion parameters. □

We remark that the proof of the expansion properties only relies on the fact that the stationary distribution of node positions is almost uniform. In fact we can get the

same expansion properties for any mobility model yielding a stationary distribution of node position that is uniform or almost uniform. Several relevant mobility models enjoy this *uniformity* property. So, thanks to our Theorem 4, we can get an upper bound on flooding time similar to that of Theorem 8.
The next theorem shows a lower bound on the flooding time of stationary geometric-MEGs.

Theorem 9 *Let $\mathcal{G}(n, \rho, r, \varepsilon)$ be a stationary geometric-MEG. If $\varepsilon \leqslant 1$, then the flooding time of $\mathcal{G}(n, \rho, r, \varepsilon)$ is w.h.p.*

$$\Omega\left(\frac{\sqrt{n}}{r+\rho}\right)$$

Idea of the Proof There are two nodes at distance $\Omega\left(\sqrt{n}\right)$ w.h.p. Take one of them as source node and observe that at every time step the *information* can get closer to the destination only for an additive factor $r + \rho$. □

Under the very reasonable conditions of the above theorem, the general bound on the flooding time of Markovian evolving graphs thus turns out to be asymptotically tight for stationary geometric-MEG.

19.3.5 Stationary Geometric-MEGs Under the Connectivity Threshold

The upper bound on flooding time in the previous section is achieved thanks to the expanding properties of the *connected* snapshots of the geometric-MEG which are, in turn, guaranteed by two facts: (1) the stationary node distribution at every time step is almost uniform and (2) the transmission radius r is over the connectivity threshold. In particular, they imply that starting from the second time step, the number of informed nodes is large enough to apply standard Chernoff-like bounds. This allowed us to evaluate the number of *new* informed nodes at any successive step. The role of node mobility is thus shown to have a negligible impact on the flooding process.

That scenario is no longer true when the transmission radius r is below the connectivity threshold (say constant). This can be viewed as a model for opportunistic MANETS where mobile nodes, in order to save energy, perform message transmissions at a relatively *low rate*. The resulting unit time (i.e., the time interval between two consecutive message transmissions) is large enough so that the mobility parameter ρ is greater than the transmission radius r.
When $\rho \gg r$, the flooding process is mainly due to *node mobility* that, roughly speaking, brings the source information *outside* the small connected components of the sparse snapshots. The next theorem provides an analytical statement of this phenomenon.

Theorem 10 *Let $\mathcal{G}(n, \rho, r, \varepsilon)$ be a stationary geometric-MEG. If $r \geqslant r_0$ and $\rho \geqslant c\sqrt{\log n}$ for sufficiently large constants r_0 and c, then the flooding time is w.h.p.*

$$O\left(\frac{\sqrt{n}}{\rho} + \log n\right)$$

Idea of the Proof The proof consists of a probabilistic analysis of the number of new informed nodes at every time step of the flooding process. In order to cope with this analysis, the temporal process is organized in three consecutive *phases*. Even though it is likely that in the real process these phases happen simultaneously rather than consecutively, our analysis yields the desired upper bound. The phases depend on the current number of informed nodes and on the "locality degree" of the process. As for the latter, we need to partition the square into equal *supercells*, i.e., subsquares of area $\Theta(\rho^2)$. This partition guarantees that any node v in a supercell S, after the move-action, can reach *any* position in *any* neighboring supercell with *almost-uniform* probability. Another crucial property yielded by the partition is that for the first—say—$O(n)$ time steps, every supercell will contain $\Theta(\rho^2)$ nodes, w.h.p.
The Bootstrap Phase. In this initial phase, we start our analysis focusing on what happens inside the neighborhood of the supercell S_0 containing the source, i.e., the supercell set $N(S_0)$ formed by S_0 and its adjacent supercells. We can say that with positive-constant probability, S_0 contains $\Theta(r^2)$ informed nodes after the first time step. Observe that this is the crucial analysis point where we need to go from *positive-constant probability* to *high probability* and we cannot use Chernoff-like bounds. Indeed, in the successive time steps $t > 0$ of this phase, we consider the flooding rate inside the supercell S'_t having the maximal number of informed nodes at time step t. We will then prove that after $O(\log n)$ time steps, there will be (at least) one supercell *quasi-informed* w.h.p., i.e., a supercell containing $\Theta(\rho^2)$ informed nodes.
The Spreading Phase. After the Bootstrap there is (at least) one quasi-informed supercell w.h.p. We can thus look at the flooding from a quasi-informed supercell to its adjacent ones. We show that if a supercell is quasi-informed at a given time step, then all its adjacent supercells will be quasi-informed within the next time step w.h.p. Since the *boundary* of any supercell set D has size at least $\Omega\left(\sqrt{|D|}\right)$, it turns out that this flooding phase makes all the supercells quasi-informed within $O\left(\sqrt{n}/\rho\right)$ time steps w.h.p.
The Filling Phase. At the end of the previous phase, we thus have all supercells quasi-informed w.h.p. The Filling phase consists of the sequence of time steps required to get all supercells informed. We prove that this final process is completed in $O(\log n)$ time steps w.h.p. □

Consequences of the bounds The results on stationary geometric-MEGs are summarized in Table 19.2.
In general, our upper bound in Theorem 10 says that in this case, *the flooding time does not asymptotically depend on the transmission radius*.

Table 19.2 Flooding time of stationary geometric-MEG $\mathcal{G}(n, \rho, r)$

$r > \sqrt{\log n}$	$\rho > \sqrt{\log n}$ and $r = \Omega(1)$
$O\left(\frac{\sqrt{n}}{r} + \log\log r\right)$	$O\left(\frac{\sqrt{n}}{\rho} + \log n\right)$
$\Omega\left(\frac{\sqrt{n}}{r+\rho}\right)$	

This fact has important technological consequences in the futuristic scenario of large, high-mobile MANETS. The two major goals in MANETS are (i) guarantee good and fast data communication and (ii) minimize node energy consumption (which is clearly an increasing function of r). It is well known that in classic (static or low-mobile) MANETS such two goals are in contrast with each other and, thus, a suitable trade-off must be determined (actually, optimizing this trade-off is currently a major research issue in ad hoc networking [2, 32, 41]). In particular, we know that [29, 37] in order to guarantee global connectivity (and thus data communication) in static random geometric graphs, the transmission radius must be $\Omega\left(\sqrt{\log n}\right)$, so it must *increase with the network size*.

In this context, our bound is a strong mathematical evidence of the fact that when node mobility is relatively high and random, the two above goals are not competing anymore. We can achieve fast data forwarding by using small transmission radius (so, saving node energy). More importantly, the transmission radius can be an *absolute constant* and, so, it does not need to increase as the network size does. The technology of node transmitter devices can be thus *scalable*. Observe that node mobility in such opportunistic networks is due to the *host* mobility which is often fully independent of sensor devices: high sensor mobility does not (necessarily) imply high energy consumption [36].

19.4 Radio Broadcasting in Dynamic Networks

In this section we study the *radio broadcast* task [10] in dynamic networks. According to the series of previous theoretical works [4, 10–12, 20, 39, 40], the communication is assumed to be synchronous. Synchronous communication allows us to focus on the impact of the *interference* phenomenon on the network performance. When a node sends a message, the latter is sent in parallel on all outgoing edges. On the other hand, a node can receive a message during a time step if and only if there is exactly one of its in-coming neighbors that sends the message during that time step. If two or more neighbors send a message during the same time step, then a *collision* occurs and the node receives nothing because of the interference phenomenon. It follows that broadcast protocols for the radio model must cope with message collisions. For instance, the flooding mechanism clearly does not work on general graphs.
In order to model a dynamic network, we use our general framework of evolving graphs. We consider two *extremal* cases: In the next section we analyze the radio

broadcast problem when the network dynamics is governed by an adaptive adversary, in Sects. 19.4.2 and 19.4.3 we consider the case in which the network dynamics is given by a sequence of independent Erdös–Rényi random graphs.

19.4.1 The Worst-Case Evolving Graph

Given a set of nodes V, we consider an *adversary* $\mathcal{A}$ that, at every time step t, chooses the set of edges $E_t \subseteq \binom{V}{E_t}$: this yields an evolving graph $\mathcal{G}_{\mathcal{A}} = \{G_t : t \in \mathbb{N}\}$. The adversary has complete knowledge of the broadcasting protocol and it is *adaptive*: at every time step t he knows the set $I_t \subseteq V$ of the informed nodes. However, the adversary must be "meaningful." An adversary is *meaningful* if, at any time step, it keeps at least one link on from an informed node to a non-informed one. The next theorem shows a simple randomized protocol that completes radio broadcasting for any choice of the meaningful adversary. We then prove that this simple protocol is asymptotically optimal.

Theorem 11 *A randomized protocol exists that, for any adaptive worst-case adversary, completes broadcasting within* $O\left(\frac{n^2}{\log n}\right)$ *time steps, w.h.p.*

Idea of the Proof Consider the following protocol: *At every time step every informed node sends the message with probability* $p = \dfrac{\log n}{n}$. Since the adversary is meaningful, at every time step a non-informed node u exists that is connected to $k \geqslant 1$ informed nodes. Since each one of those nodes sends the message independently with probability $\log n/n$, node u gets the message with probability larger than $\log n/n$. □

It is important to observe that *no deterministic* protocol can complete broadcast against meaningful adversaries. So, the above theorem shows another relevant case in distributed computing where random computations are provably more powerful than deterministic ones.
The next theorem shows that for every protocol we can define an adaptive meaningful adversary so that the resulting expected completion time is $\Omega\left(\frac{n^2}{\log n}\right)$.

Theorem 12 *For every randomized broadcast protocol, an adaptive worst-case adversary exists that forces the protocol to have* $\Omega\left(\frac{n^2}{\log n}\right)$ *expected completion time.*

Idea of the Proof Consider the following adversarial strategy. At every time step, choose an arbitrary non-informed node v; if a node u exists that is transmitting with probability $p_u \leqslant \log m/m$, where m is the number of informed nodes in the current time step, then connect node u with v and leave all the other edges down. Otherwise connect to v all the informed nodes and leave all the other edges down. In both cases the probability that node v gets the information is at most $\log m/m$; hence the expected number of time steps required to have one newly informed node is at least $\frac{m}{\log m}$. This implies that the expected completion time of the protocol is at least

$$\sum_{m=2}^{\infty} \frac{m}{\log m} = \Omega\left(\frac{n^2}{\log n}\right)$$

□

19.4.2 The Random Evolving Graph: Case p Known

For any n and for any probability p, the random evolving graph, denoted as *dynamic* $G_{n,p}$, is an infinite sequence of random graphs $G_0, G_1, \ldots, G_t, \ldots$ where each G_t is independently selected according to the *random graph model* $G_{n,p}$. In the sequel p will denote the edge probability of random graphs. A broadcast protocol in dynamic $G_{n,p}$, at any time step t, acts in graph G_t. Notice that this is a special case of edge-Markovian evolving graph (see Definition 1 with $q = 1 - p$).
We now present an oblivious randomized protocol that makes use of an oblivious version (the third loop below) of the BGI's *Decay* procedure [4].

```
DynBroad(n,p)

for ⌈$c \log n$⌉ time steps (where $c$ is a suitable constant)
        The source sends the message;
for ⌈$c \log n$⌉ time steps
        Each informed node sends the message;
for $k = 0, 1, \ldots \lceil \log n \rceil$
        Each informed node sends the message with probability $q = e^{-k}$
for ⌈$c \log n$⌉ time steps
        Each informed node sends the message with probability $q = 1/(np)$
```

The protocol clearly terminates within $O(\log n)$ time steps. In what follows we show that for $p \geqslant 1/n$, `DynBroad(n,p)` completes broadcasting in dynamic $G_{n,p}$, w.h.p. The proof evaluates the number of informed nodes after each of the four loops of the protocol. Note that the analysis significantly departs from those in [4] and [24] for static unknown graphs.

Theorem 13 *Let $p \geqslant 1/n$. Protocol `DynBroad(n,p)` completes broadcasting in dynamic $G_{n,p}$, w.h.p.*

Idea of the Proof Phase 1 (First loop) In this phase there are no collisions, because only the source node is transmitting. A node u receives the message if and only

if a time step exists such that u is connected to the source node. At every time step the link between u and the source is up with probability p independently of the other time steps; hence the probability that node u will never be connected to the source in this phase is $(1-p)^{\Theta(\log n)} \approx e^{-\Theta(p \log n)}$. The expected number of informed nodes at the end of the phase is thus $n\left(1-e^{-\Theta(p \log n)}\right)$. If $p \geqslant 1/\log n$ this is a constant fraction of n, while if $1/n \leqslant p \leqslant 1/\log n$ it holds that $n\left(1-e^{-\Theta(p \log n)}\right) \approx np \log n \geqslant \log n$. In both cases we can apply Chernoff bounds to obtain concentration results. Thus, if $p \geqslant 1/\log n$, then after the first phase a constant fraction of nodes is informed w.h.p., and we can directly jump to the analysis of the fourth phase; while if $1/n \leqslant p \leqslant 1/\log n$, after the first phase there are at least $\log n$ informed nodes w.h.p. and we proceed with the analysis of the second and third phases.
Phase 2 (Second loop) Our aim is to show that at the end of this phase, at least $1/p$ nodes are informed w.h.p. We may assume that we start with at least $\log n$ informed nodes (because of the previous phase), and that $1/n \leqslant p \leqslant 1/\log n$ (because otherwise we can directly go to phase 4).

In this phase all the informed nodes send the message at every time step; hence a node u, that is not informed at the beginning of the phase, receives the message if and only if a time step exists such that u is connected to exactly one of the informed nodes in such time step.
Let t be an arbitrary time step, let m_t be the number of informed nodes in such time step, and let u be a non-informed node at time step t. If $m_t \geqslant 1/p$ we have done; otherwise the probability that u receives the message in the current time step is $m_t p(1-p)^{m_t-1}$. Since by hypothesis $m_t \leqslant 1/p$ and $p \geqslant 1/n$ we have that $m_t p(1-p)^{m_t-1}$ is approximately at least as large as $m_t p \geqslant m_t/n$. The expected number of new informed nodes in the current time step is thus at least $(n-m_t)m_t/n \geqslant \alpha m_t$ for a suitable positive constant α. By using Chernoff bound, since $m_t \geqslant \log n$, once again this holds w.h.p.
Hence, in every time step t of this phase, either the number of informed nodes m_t is already larger than $1/p$ or the recurrence $m_t \geqslant (1+\alpha)m_t$ holds w.h.p. for some positive constant α. By iterating the recurrence for $O(\log n)$ time step it follows that at the end of the phase there must be at least $1/p$ informed nodes w.h.p.
Phase 3 (Third loop) Now we show that if we start the third loop with at least $1/p$ informed nodes, then at the end of the phase a constant fraction of nodes will be informed w.h.p.
At time step t of this phase every informed node sends the message with probability $q_t = e^{-t}$; hence the probability that a non-informed node u receive the message in such time step is $m_t p q_t \,(1-pq_t)^{m_t-1}$. Since q_t decreases exponentially from 1 to $1/n$, and since m_t can only increase, a time step t must exist such that $m_t p$ is close to $1/q_t$ up to a constant factor. In that time step, every non-informed node has constant probability to be informed. Hence, after that, the total number of informed nodes is at least a constant fraction of n.
Observe that we cannot directly use Chernoff bounds here to achieve concentration results: the involved random variables are not independent. However, by using a suitable *conditioning trick* (we refer the interested reader to the full proof in [16, 17])

it is possible to prove that the number of informed nodes at the end of the phase is at least a constant fraction of n, w.h.p.

Phase 4 (Fourth loop) At the beginning of this phase there are at least a constant fraction of informed nodes. If every informed node sends the message with probability $1/np$, then every non-informed node u has a constant probability to be informed in every time step. Hence node u will be informed by the end of the phase w.h.p. By using union bound, it follows that every node will be informed by the end of the phase w.h.p. □

We observe that if p is $1 - o(1)$, the broadcast task can be completed in $o(\log n)$ by considering the simple protocol where only the source transmits with probability 1 (e.g., if $p = 1 - 1/n^2$ broadcasting is completed w.h.p. in one time step). Instead when p does not tend to 1 as n goes to infinity, we now show a lower bound matching the previous logarithmic upper bound.

Theorem 14 *Let ε be any positive constant and let $p \leqslant 1 - \varepsilon$. Any broadcast protocol in dynamic $G_{n,p}$ has expected completion time $\Omega(\log n)$.*

Idea of the Proof Let m_t be the number of informed nodes at time step t. Assume k nodes are transmitting at time step t, with $0 \leqslant k \leqslant m_t$. The probability that a non-informed node u receives the information is thus $kp(1-p)^{k-1}$ which is less than some constant $\delta < 1$ for every k if $p \leqslant 1 - \varepsilon$. Hence the expected number of new informed nodes cannot be larger than a δ-fraction of all the non-informed nodes. In order to get all the nodes informed, the protocol thus needs $\Omega(\log n)$ time steps. □

Protocol `DynBroad(n,p)` requires the knowledge of the edge probability p. In the next section we investigate what happens when this is not the case.

19.4.3 The Random Evolving Graph: Case p Unknown

Let us consider the following variant of the BGI's Decay procedure [4] denoted as `BGI(n)`.

```
BGI(n)

for ⌈c log n⌉ time steps (where c is a suitable constant)
    for k = 0, 1, ... ⌈log n⌉
        Each informed node sends the message with probability $q = e^{-k}$
```

Protocol `BGI(n)` terminates within $O(\log^2 n)$ time steps. Now we show that it completes broadcasting in dynamic $G_{n,p}$, w.h.p. Through the following we call *phase* each execution of the inner **for** of the protocol `BGI(n)`.

Theorem 15 *Protocol `BGI(n)` completes broadcasting in dynamic $G_{n,p}$ w.h.p. for any $p \geqslant 1/n$.*

Idea of the Proof The proof proceeds as follows: first we show that after $O(\log^2 n)$ time step there are at least $\Omega(\log n)$ informed nodes. Then observe that protocol `BGI(`n`)` *simulates* phases 2, 3, and 4 of protocol `DynBroad(`n, p`)`. Indeed, (i) a subsequence of $\Theta(\log n)$ time steps exists in which every informed node sends the message (whenever $k = 0$, protocol `BGI(`n`)` simulates phase 2 of `DynBroad(`n, p`)`); (ii) the third loop of `DynBroad(`n, p`)` is just one of the inner loops of `BGI(n)`; (iii) a subsequence of $\Theta(\log n)$ time steps exists in which every informed node sends the message with probability close to $1/(np)$ (whenever $k \approx \log(np)$, protocol `BGI(`n`)` simulates phase 4 of `DynBroad(`n, p`)`). Hence, even though those subsequences are made by non-consecutive time steps, it is possible to prove the thesis by adapting the strategy used in the proof of Theorem 13. □

A protocol is said *homogeneous* if at every time step every informed node follows the same *rule*. Protocol `BGI(n)` is thus homogeneous because at every time step every informed node transmits with the same probability, while protocol `DynBroad(n,p)` is not homogeneous, because in its first phase the source is the only one sending the message.

For homogeneous protocols, we can give a lower bound that *almost* matches the completion time of `BGI(n)`. In the next theorem we show that when a homogeneous randomized protocol does not know p, an adversary can choose p in order to force the protocol to run for $\Omega(\log^2 n/\log\log n)$ expected time.

Theorem 16 *Given any homogeneous broadcast protocol $\mathcal{P}$, a probability p exists such that $\mathcal{P}$ has expected completion time $\Omega\left(\frac{\log^2 n}{\log\log n}\right)$ in dynamic $G_{n,p}$.*

Idea of the Proof For any fixed edge probability p, there exists an interval of transmission probabilities such that if the protocol's transmission probability is out of this interval then the number of new informed nodes is *small*. There exist $\Omega\left(\frac{\log n}{\log\log n}\right)$ edge probabilities such that their corresponding intervals are pairwise disjoint. A homogeneous broadcast protocol that does not know the probability p of the dynamic $G_{n,p}$ cannot avoid that at least one of these intervals (and the corresponding edge probability $\tilde{p}$) does exist that contains at most $O(\log n)$ transmission probabilities of the protocol. Hence, for most of the time steps in dynamic $G_{n,\tilde{p}}$, the number of new informed nodes will be small. □

It is an interesting open question whether a *non-homogeneous* protocol exists that *beats* this lower bound. Any positive answer to this question would be very surprising in such dynamic and unknown scenario.

19.4.3.1 Consequences of the bounds

The analysis of the *worst-case* evolving graphs shows that *meaningful* network dynamics cannot arbitrarily slow down the information propagation. In particular, our quadratic upper bound implies that no meaningful adversary exists that forces exponential broadcast completion time. On the other hand, the analysis of the dynamic $G_{n,p}$ evolving graph shows that, if the network evolution is somewhat *random*, then such dynamic environment can allow fast broadcast completion time even on networks that are very sparse and disconnected at every time step (e.g., dynamic $G_{n,p}$ with $p \approx 1/n$).

Throughout the chapter we used the expression w.h.p. as a shortcut to say that an event holds with probability at least $1 - 1/n^{\alpha}$ for some positive constant α. For example, we showed the probability that protocols `DynBroad(n,p)` and `BGI(n)` do not complete the broadcast task is less than $n^{-\alpha}$ for some positive constant α. It may be worth noticing that constant α depends on constant "c" that appears in the protocols. By choosing a sufficiently large c, we can thus make the probability of failure polynomially small while keeping the same asymptotic running time.

19.5 Conclusions and Open Problems

In this chapter we analyzed the speed of information spreading in dynamic networks by using the general framework of evolving graphs and simple communication procedures like *flooding*. The flooding time plays the same role for evolving graphs as the diameter does for static graphs, and it is thus a natural lower bound on the completion time of *any* communication procedure in dynamic networks.

A comparison of the results on flooding time (Theorems 1, 2, and 3) with the ones on radio broadcast time (Theorems 13 and 14) in the dynamic $G_{n,p}$ model highlights the *impact of collisions* on the speed of information spreading. When the random evolving graph is very sparse and disconnected ($p \approx 1/n$) both flooding time and radio broadcast time are logarithmic, but when p grows and the graph becomes denser and denser, the flooding time drops down fast to a constant value, while the radio broadcast time stays logarithmic for the whole range of p, slacken by the interference phenomenon.

Several open challenging issues in dynamic sensor networks are still far to be well formalized and studied from a foundational point of view. Among them, we briefly discuss the ones we believe more relevant and more suitable to be addressed by our approach.

Modeling faulty sensor networks by a more *concrete* class of Markovian evolving graphs still keeping its analytical study a possible task represents one of the most important open challenges in this area. In particular, a good direction could be that of introducing stochastic dependence among faults in the same local area.

Our method provides "good" upper bounds in Markovian evolving graphs having an *almost homogeneous* topology. Another important issue is to investigate evolving graphs that are somewhat *non-homogeneous*. For instance, we can consider mobility models yielded by node random walks over highly irregular support graphs. A further instance is that yielded by the *random waypoint* model over a non-convex, irregular region.

Finally, it would be interesting to extend our analysis to other basic communication tasks such as data gathering and routing.

References

1. D. Aldous and J. Fill. *Reversible Markov Chains and Random Walks on Graphs*. http://stat-www.berkeley.edu/users/aldous/RWG/book.html, 2002.
2. C. Ambühl. An optimal bound for the MST algorithm to compute energy efficient broadcast trees in wireless networks. In: *Proceedings of 32th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3580 of *LNCS*, pages 1139–1150. Springer, Berlin Heidelberg, 2005.
3. C. Avin, M. Koucky, and Z. Lotker. How to explore a fast-changing world. In *Proceedings of 35th International Colloquium on Automata, Languages and Programming (ICALP'08)*, volume 5125 of *LNCS*, pages 121–132. Springer, Berlin Heidelberg, 2008.
4. R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45(1):104–126, 1992.
5. H. Baumann, P. Crescenzi, and P. Fraigniaud. Parsimonious flooding in dynamic graphs. In *Proceedings of 28th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 260–269. ACM Press, New York, NY, USA, 2009.
6. S. Bhadra and A. Ferreira. Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks. In: *Proceedings of the Second International Conference, ADHOC-NOW*, volume 2865 of *LNCS*, pages 259–270. Springer, Berlin Heidelberg, 2003.
7. B. Bollobás. *Random Graphs*. Cambridge University Press, Cambridge, 2001.
8. B. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2):267–285, 2003.
9. T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communication and Mobile Computing*, 2(5):483–502, 2002.
10. I. Chlamtac and S. Kutten. On broadcasting in radio networks - problem analysis and protocol design. *IEEE Transactions on Communications*, 33(12):1240–1246, 1985.
11. B. Chlebus, L. Gasieniec, A. Gibbons, A. Pelc, and W. Rytter. Deterministic broadcasting in unknown radio networks. In: *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 861–870. ACM Press, New York, NY, USA, 2000.
12. M. Chrobak, L. Gasieniec, and W. Rytter. Fast broadcasting and gossiping in radio networks. *Journal of Algorithms*, 43(2):177–189, 2002.
13. F. Chung and L. Lu. The diameter of sparse random graphs. *Advances in Applied Mathematics*, 26:257–279, 2001.
14. A. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri. Flooding time of edge-markovian evolving graphs. SIAM Journal on Discrete Mathematics (accepted).
15. A. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri. Flooding time in edge-markovian dynamic graphs. In: *Proceedings of 27th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 213–222. ACM Press, New York, NY, USA, 2008.
16. A. Clementi, A. Monti, F. Pasquale, and R. Silvestri. Communication in dynamic radio networks. In: *Proceedings of 26th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 205–214. ACM Press, New York, NY, USA, 2007.
17. A. Clementi, A. Monti, F. Pasquale, and R. Silvestri. Broadcasting in dynamic radio networks. *Journal of Computer and System Sciences*, 75(4):213–230, Academic Press, Inc., Orlando, FL, USA, 2009.
18. A. Clementi, A. Monti, F. Pasquale, and R. Silvestri. Information spreading in stationary markovian evolving graphs. In: *Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium*, pages 1–12, IEEE Computer Society, 2009.
19. A. Clementi, F. Pasquale, and R. Silvestri. Manets: High mobility can make up for low transmission power. In: *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP'09)*, volume 5556 of *LNCS*, pages 387–398. Springer, 2009.

20. A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology. *Journal of Algorithms*, 60(2):115–143, 2006.
21. J. Diaz, D. Mitsche, and X. Perez-Gimenez. On the connectivity of dynamic random geometric graphs. In: *Proceedings of 19th annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 601–610, ACM Press, New York, NY, USA, 2008.
22. J. Diaz, X. Perez, M.J. Serna, and N.C. Wormald. Walkers on the cycle and the grid. *SIAM Journal on Discrete Mathematics*, 22(2):747–775, 2008.
23. T. Dimitriou, S. Nikoletseas, and P. Spirakis. The infection time of graphs. *Discrete Applied Mathematics*, 154(18):2577–2589, 2006.
24. R. Elsässer and L. Gasieniec. Radio communication in random graphs. *Journal of Computer and System Sciences*, 72(3):490–506, 2006.
25. P. Erdös and A. Rényi. On the evolution of random graphs. *Publications Mathematical Institute Hungarian Academy Science*, 5:17–61, 1960.
26. A. Ferreira. Building a reference combinatorial model for manet. *IEEE Network*, 18(5):24–29, 2004.
27. M. Grossglauser and N.C. Tse. Mobility increases the capacity of ad-hoc wireless networks. *IEEE/ACM Transactions on Networking*, 10(4):477–486, 2002.
28. R.A. Guerin. Channel occupancy time distribution in a cellular radio system. *IEEE Transactions on Vehicular Technology*, 36(3):89–99, 1987.
29. P. Gupta and P.R. Kumar. Critical power for asymptotic connectivity in wireless networks. *Stochastic Analysis, Control, Optimization and Applications*, pages 547–566, 1998.
30. A. Jardosh, E.M. Belding-Royer, K.C. Almeroth, and S. Suri. Towards realistic mobility models for mobile ad hoc networks. In: *Proceedings of the 9th annual international conference on Mobile computing and networking (Mobicom)*, pages 217–229, ACM Press, New York, NY, USA, 2003.
31. D. Kempe and J. Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In: *Proceedings of 43rd Symposium on Foundations of Computer Science (FOCS)*, pages 471–480, IEEE Computer Society, Washington, DC, USA, 2002.
32. L. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc. Power consumption in packet radio networks. *Theoretical Computer Science*, 243:289–305, 2000.
33. J.-Y. Le Boudec and M. Vojnovic. Perfect simulation and the stationarity of a class of mobility models. In: *Proceedings of 24th IEEE INFOCOM*, pages 2743–2754, IEEE Computer Society, Washington, DC, USA, 2005.
34. S. Nikoletseas, C. Raptopoulos, and P. Spirakis. The survival of the weakest in networks. *Computational and Mathematical Organization Theory*, Kluwer Academic Publishers, Hingham, MA, USA, 2008.
35. R. O'Dell and R. Wattenhofer. Information dissemination in highly dynamic graphs. In: *Proceedings of ACM DIALM-POMC*, pages 104–110, ACM Press, New York, NY, USA, 2005.
36. L. Pelusi, A. Passarella, and M. Conti. Beyond manets: Dissertation on opportunistic networking. *IIT-CNR Technical Report*, 2006.
37. M. Penrose. *Random Geometric Graphs*. Oxford University Press, New York, 2003.
38. B. Pittel. On spreading a rumor. *SIAM Journal on Applied Mathematics*, 47(1):213–223, 1987.
39. S. Ramanathan and E. Lloyd. Scheduling algorithms for multihop radio networks. *IEEE/ACM Transactions on Networking*, 1(2):166–177, 1993.
40. R. Ramaswami and K. Parhi. Distributed scheduling of broadcasts in radio network. In *Proceedings of IEEE INFOCOM*, IEEE Computer Society, Washington, DC, USA, pages 497–504, 1989.
41. P. Santi and D. M. Blough. The critical transmitting range for connectivity in sparse wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):25–39, 2003.
42. J. Svante, T. Luczak, and A. Rucinski. *Random Graphs*. Wiley-Interscience, New York, 2000.

Chapter 20
Self-Stabilizing and Self-Organizing Virtual Infrastructures for Mobile Networks

Shlomi Dolev and Nir Tzachar

Abstract Self-stabilizing algorithms can be started in any arbitrary state to exhibit a desired behavior following a convergence period. The class of self-organizing distributed algorithms is regarded here as a subclass of the self-stabilizing class of algorithms, where convergence is sub-linear in the size of the system and local perturbation of state is handled locally converging faster than the convergence from an arbitrary state. The chapter starts with a short overview of several virtual infrastructures and fitting self-stabilizing and self-organizing techniques:

- Group communication by random walks [23]
- Polygon-based stateless infrastructure [19]
- Geographic quorum systems [1, 16]
- Autonomous virtual node [18]
- Secret swarm units [20]
- Spanners, spanning expanders

The last design, which is based on expanders and short random walks, is described in detail.

20.1 Introduction

In the scope of sensor networks, traditional paradigms and techniques for forming communication infrastructure among computing devices must be reexamined. In particular, the self-stabilization [9], [10] property, an important property of any dynamic long-lived system, must be taken into account. Self-stabilizing systems may start operating in any arbitrary state and can therefore recover following a temporary violation of the assumption made by the system designer. Mobile ad hoc networks are very dynamic in nature and must cope with unreliable and sometimes unpredictable environments. Thus, the design of self-stabilizing mobile and ad hoc

S. Dolev (✉)
Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, 84105, Israel
e-mail: dolev@cs.bgu.ac.il

Partially supported by EU ICT-2008-215270 FRONTS, Rita Altura Trust Chair in Computer Sciences, and the Lynne and William Frankel Center for Computer Sciences.

networks is of great importance. Self-stabilizing networks are *self-organizing* if they are very efficient in terms of the time it takes for the system to converge. Namely, self-stabilizing and self-organizing systems start to operate as they should in sub-linear time, and once they start to operate correctly, they react to dynamic changes even faster. We overview several recent works demonstrating several directions for creating adaptive infrastructures and abstractions, namely, self-stabilizing and self-organizing infrastructures. These infrastructures fit the mobile ad hoc network characteristic.

This chapter is devoted to several self-stabilizing and self-organizing infrastructures; where one of them, which is based on expanders and short random walks, is described in detail (Sect. 20.2 and onward).

- **Group Communication by random walks [23]**. Random walk forms an important tool to cope with the frequent changes of dynamic networks, as it does not need to maintain and update a distributed (routing) data structure. Using random walks for implementing group communication abstractions is suggested in [23]. The system design is based on a mobile agent, collecting and distributing information, during a random walk. Three possible settings for modeling the location of the processors in the ad hoc network are presented:

  - *Slow location change:* Where the random walk is analyzed as if acting on a fixed network. Thus, the expected cover time in terms of agent moves is $O(n^3)$, where n is the number of nodes in the network.
  - *Complete random change:* Where a reduction to a random walk in the complete graph case is appropriate. In this case the expected cover time in terms of agent moves is $O(n \log n)$.
  - *Neighbors with probability:* Where one may tune the random walk by the probability differences to mimic equal probability transfer, which is in fact a reduction to the case of a fixed network.

  The new techniques that are based on random walks support group membership and multicast and also support resource allocation. To support group membership the agent maintains a list of members with the number of steps elapsed since they were visited. Since the system should be self-stabilizing, we must consider cases in which the list of members carried with the agent totally does not correspond to the members of the group. Thus, members that are not visited during a too long period, where period is measured in terms of number of agent steps, are removed. Moreover, if the sorted list of last visiting times, $v_1, v_2, \ldots, v_m$, indicates that v_{i+1} is much larger than the cover time of a component with $i + 1$ nodes, then all members but the first i are suspected as a result of a corruption or a dynamic change and are excluded from the group.

- **Polygon-based stateless infrastructure [19].** Another approach to create an infrastructure without maintaining a distributed data structure, or in other words to use stateless routing suggested in [19], where the idea is to form routing information on the fly. Consider communication among sensors that are deployed in a geographic region. Each sensor is a computing device with severe resource limitations, low power, slow processing, and small memory. The devices are distributed (uniformly) in the geographic region. Imaginary polygon tilings are presented as a general scheme for supporting communication in sensor networks. The tiling system is built on the fly, ensuring that most of the sensors do not have to transmit. Assume that a certain sensor that is planning to broadcast a message (a) chooses a coordinate system in which it is in the center of a tile; (b) locally broadcasts the message to the sensors in its tile; (c) notifies sensors in the centers of the neighboring tiles that they should locally broadcast the message to their tile and forward the message to the center of their neighboring tiles. This way, eventually local transmissions are received in the entire geographic region. Definition of virtual tiles is used to implement self-stabilizing broadcast, flooding, and sense of direction procedures that fit the special characteristics of the system. In addition, the tiles communication primitives allow a scheme for distributing secrets that activate the sensors simultaneously at a particular time without revealing the nature of the upcoming activity.

- **Geographic quorum systems [1, 16].** Having a fixed infrastructure for ad hoc networks seems an impossible task. An approach that is based on associating object with certain geographic locations is suggested in [1, 16]. This new approach, called the GeoQuorums approach, supports implementing atomic read-/write shared memory in mobile *ad hoc* networks. The existence of *focal points*, geographic areas that are normally "populated" by mobile nodes, is assumed. For example, a focal point may be a road junction, a scenic observation point, or a water resource in the desert. Mobile nodes that happen to populate a focal point participate in implementing a shared atomic object, using a replicated state machine approach. When a mobile node joins a geographic region, the mobile node receives a copy of the memory from the nodes in the region; when the node leaves the region the node does not have to maintain the copy anymore. A read request from an object that resides in a certain region may be answered by any of the nodes that populate the region; a write is an update of the copies of all the nodes in the region. These objects, which are called focal point objects, are then used to implement atomic read/write operations on a virtual shared object, using the GeoQuorums algorithm. The GeoQuorums algorithm uses a quorum-based strategy in which each quorum consists of a set of focal point objects. The quorums are used to maintain the consistency of the shared memory and to tolerate limited failures of the focal point objects, caused by depopulation of the corresponding geographic areas.

- **Autonomous virtual node [18].** The next step is to allow the object to act as an automaton rather than only memory. Consider the case where each mobile node is a car, assuming further that the focal point is a road junction. One may implement a virtual traffic light using the focal point abstraction. When a car arrives to the junction it starts the virtual traffic light with green light in its direction. Cars that arrive to a junction that is already populated will copy the state of the traffic light from the other cars and continue executing the (common) program of the traffic light. One may even let the virtual object move according to its own decisions (scan the traffic jam using the communication capabilities of the cars and transfer summary information, possibly information on the length of the jam). A new abstraction for virtual infrastructure in mobile ad hoc networks is presented in [18]. An autonomous virtual mobile node (AVMN) is a robust and reliable entity that is designed to cope with the inherent difficulties caused by processors arriving, leaving, and moving according to their own agendas, as well as with failures and energy limitations. The AVMN is a virtual general-purpose computing entity, an automaton that can make autonomous on-line decisions concerning its own movement. A self-stabilizing implementation of this new abstraction that provides automatic recovery from any corrupted state is presented in [18].

- **Secret swarm units [20].** Security considerations must be a part of network design (e.g., [15]). Secret sharing is a fundamental cryptographic task. Motivated by the virtual automata abstraction and swarm computing, an extension of the k-secret sharing scheme is investigated in [20]; in the suggested extension the secret shares are changed on the fly, independently, and without (internal) communication as a reaction to a global external trigger. The changes are made while maintaining the requirement that k or more secret shares may reconstruct the secret and no $k - 1$ or fewer can do so. The application considered is a swarm of mobile processes, each maintaining a share of the secret which may change according to common outside inputs, e.g., inputs received by sensors attached to each process. The proposed schemes support the addition and removal of processes from the swarm, as well as corruption of a small portion of the processes in the swarm. There are three approaches, one is based on secret sharing and the linearity of operations on the shares, thus supporting addition and multiplication by constants. The swarm members simultaneously receive the constant and add or multiply the constant to the shares, and by doing so, the secret is modified in the same manner. Interestingly, no communication among the swarm members is required. The second scheme is based on the Chinese remainder theorem, encoding the secret by different components of the counter value computed modulo distinct prime numbers. The last scheme encodes the actual state of the swarm as the state that appears the most in the copy maintained by the members. This approach, though it may leak some information on the current state of the swarm, allows the implementation of any automaton. Support for cases of partial corruption is suggested for any of the schemes. The use of an efficient multi-party computation [11, 12] may process an infinite sequence of inputs using one round of communication to perform a step of a universal Turing machine.

20.2 Self-Stabilizing and Self-Organizing Distributed Algorithms

Constructing spanning infrastructures, such as spanning trees or expanders, in a way that fits the speed of convergence needed for mobile networks is presented in [24, 26]. Self-stabilization ensures automatic recovery from an arbitrary state; *self-organization* is defined as a property of algorithms which displays local attributes. More precisely, an algorithm is self-organizing if it (a) converges in sub-linear time and (b) reacts "fast" to topology changes. If $s(n)$ is an upper bound on the convergence time and $d(n)$ is an upper bound on the convergence time following a topology change, then $s(n) \in o(n)$ and $d(n) \in o(s(n))$. The self-organization property can then be used for gaining, in sub-linear time, global properties and reaction to changes. Self-stabilizing and self-organizing algorithms for many distributed algorithms, including distributed snapshot and leader election, are presented in [24]. The algorithms assume that it is possible to locally define (and then use) hyperlinks, just like phone connections. A new randomized self-stabilizing distributed algorithm for cluster definition in communication graphs of bounded degree processors is presented in [24]. These graphs reflect sensor networks deployment. The algorithm converges in $O(\log n)$ expected number of rounds, handles dynamic changes locally, and is, therefore, *self-organizing*.

20.2.1 Spanders, Spanning Expanders

Expanders are useful for the cases in which hyperlinks are not supported. In [25, 26] self-stabilizing and self-organizing distributed construction of a spanner that forms an expander is considered, namely, constructions that are extremely local, robust, and dynamic. Folklore results for randomly defining an expander are used. Given the randomized nature of the algorithms, a monitoring technique is presented for ensuring the desired results. The monitoring is based on the fact that expanders have a rapid mixing time and the possibility of examining the rapid mixing time by $O(n \cdot \log n)$ short ($O(\log^4 n)$ length) random walks even for non-regular expanders. We then employ our results to construct a hierarchical sequence of *spanders*, each of them an expander spanning the previous one. Such a sequence of spanders may be used to achieve different quality of service assurances in different applications.

20.2.2 Distributed Expander Construction, Related Work

To the best of our knowledge, there is a limited number of works that address the problem of distributed expander construction. In [31], the authors propose to construct an expander graph by composing a sufficient number of Hamiltonian cycles. The proposed construction makes the following assumptions: the communication network is an overlay network (two nodes can directly communicate as long as their identifiers are known to each other), the algorithm starts from a predefined graph of at least three nodes, and nodes wishing to join the graph must send a special message to a node that has already joined the graph. Unfortunately, the proposed algorithms cannot be started in an arbitrary state and therefore are not self-stabilizing.

A different approach for distributed construction of expanders is proposed in [35]. The authors suggest using uniform sampling to select, for each node, a set of expander-neighbors. The goal in [35] is to construct an "almost" regular graph, where each node maintains a list of expander neighbors of size between $d - c$

and $d + c$, where d is the desired degree of the almost regular graph and c is a small constant. When a node v selects a node u to be its neighbor, v sends a JOIN message to u. However, u might already have more than $d + c$ neighbors, so u throws one neighbor, chosen uniformly, from u's neighbor list and inserts v instead. u then sends a LEAVE message to the thrown neighbor, w, to notify w to remove u from w's neighbor list. As a result, it is not straightforward to predict whether the given algorithm converges or oscillates among different incomplete graphs.

The definition of self-organization we use was first presented in [24]. However, the system model of [24] differs in the present chapter; in [24] we assumed that the system is designed to support *hyperlinks*. That is, given a path between two nodes, u and v, a direct link between u and v may be established. Moreover, the communication overhead of such links is assumed to take one time unit. In contrast, our model assumes a more conventional system, in which hyperlinks cannot be defined. The algorithms we present achieve self-organization by employing the characteristics of the underlying expander graph.

Expansion evaluation has been considered in the past. For example, from a mathematical viewpoint, our techniques for expansion evaluation resemble those seen in [27]. However, the authors of [27] only deal with bounded degree graphs, and their methods are not readily converted to distributed settings. Independently (of [26]), in a recent work, Czumaj and Sohler [8] extend the result of [27], selecting optimal parameters for the expansion testing algorithm. Recently, [29, 34] have improved upon the results of [8]. The results obtained are closely related to our own. However, the question of a distributed implementation remains in all of the above.

20.3 System Settings

The *system* is defined by a *communication graph*, $G = (V, E)$, where V is a set of *nodes* $\{v_1, v_2 \ldots, v_n\}$ and E is a set of undirected *communication links*; if $(v, u) \in E$ then v and u can communicate by sending *messages* of bounded size to each other. Message sizes are restricted to $O(\log n)$ bits. We further assume that each communication link is a bounded capacity FIFO queue. Let lc denote the capacity of the links. When messages are sent over a full link, we assume that one of the messages (either already in the link, or the new one) is lost. We only require messages which are sent infinitely often and are received infinitely often. We present data-link algorithms to ensure that communication over such links is snap-stabilizing (see Sect. 20.5.3.1).

Nodes may join and leave the system at any time. We make no distinction between a node that leaves and a node that crashes, assuming that both can be detected by neighboring nodes in a timely fashion.

For a graph $G = (V, E)$, given two sets of nodes, V_1 and V_2, we define the following: $E(V_1, V_2) = \{e = (v_1, v_2) \in E | v_1 \in V_1 \wedge v_2 \in V_2\}$ (e.g., the set of edges between V_1 and V_2). We also define $\overline{V_1} = V \setminus V_1$, the set of nodes not in V_1.

A graph $G = (V, E)$ is an edge expander if there exists a constant a, such that for each set S of vertices (where $|S| < |V|/2$) it follows that $|E(S, \overline{S})|/|S| > a$. For a comprehensive overview of expander graphs, their properties and sample uses thereof, we recommend for the reader the excellent text found in [28].

Given a graph $G = (V, E)$, a *spander*, $\mathcal{S} = (V, E')$, is a spanning subgraph of G if there exists a constant $p > 0$, such that $|E'| \leq p|E|$ and the edge expansion of $\mathcal{S}$ is at worst p times the edge expansion of G.

A configuration c of the system is a tuple $c = (S, L)$; S is a vector of states, $\langle s_1, s_2, \cdots s_n \rangle$, where the state s_i is a state of node v_i; L is a vector of *link states* $\langle l_{i,j}, \cdots \rangle$ for each $(i, j) \in E$. A *link* $l_{i,j}$ is modeled by a FIFO queue of messages that are waiting to be received by v_j and the content of the queue is the state of the link. Whenever v_i sends a message m to v_j, m is enqueued in $l_{i,j}$ (if the link is full, an arbitrary message in the queue will be dropped). Also, whenever v_j receives a message m from v_i, m is dequeued from $l_{i,j}$. A node changes its state according to its transition function (or program). A transition of node v_i from a state s_j to state s_k is called an *atomic step* (or simply a step) and is denoted by a. A step a consists of local computation and terminates with either a single send or a single receive operation.

The system is asynchronous, meaning that there is no correlation between the non-constant rate of steps taken by the nodes. We model our system using the interleaving model. An *execution* is a sequence of global configurations and *steps*, $\mathcal{E} =< c_0, a_0, c_1, a_1, \ldots >$, so that configuration c_i is reached from c_{i-1} by a step a_i of one node v_j. The states changed in c_i, due to a_i, are the one of v_j (which is changed according to the transition function of v_j) and possibly the state of a link attached to v_j. The content of a link state is changed when v_j sends or receives a message during a_i. An execution $\mathcal{E}$ is *fair* if every node executes a step infinitely often in $\mathcal{E}$. Within the scope of self-stabilization we consider executions that are started in an arbitrary initial configuration.

A *task* is defined by a set of executions called *legal executions* and is denoted by LE. A configuration c is a *safe configuration* for a system and a set of legal executions LE if every fair execution that starts in c is in LE. A system is self-stabilizing for a task and a set of legal executions LE if every infinite execution reaches a safe configuration in relation to LE. We sometimes use the term "the algorithm stabilizes" to note that the algorithm has reached a safe configuration with regard to the legal execution of the corresponding task.

To measure time we use the notion of *communication rounds*: a communication round (or just a round) is a sequence of atomic steps such that each node has taken at least one atomic step during this sequence. If this atomic step involves a send operation of a message m over link l, then we require that the atomic step which corresponds to receiving a message from l, which was sent during this sequence of atomic steps, will also appear in the sequence. Such time measurements are appropriate when a protocol involves the entire system. When measuring the time complexity of a protocol which involves only a subset of nodes in the system, we use the notion of the *happened before* relation (see [30]). We then say that the time complexity of the protocol is the longest chain of happened before relation induced by the protocol.

A distributed algorithm is termed *self-organizing* ([24]) if it satisfies the following properties: (1) the algorithm is self-stabilizing, (2) convergence time to a safe configuration, $s(n)$, is in $o(n)$, and (3) after reaching a safe configuration, convergence time following a dynamic change, $d(n)$, is in $o(s(n))$.

A distributed algorithm is termed *snap-stabilizing* if the algorithm stabilizes following the first request by any node and before, or simultaneously with, a notification arriving to the requesting node at the completion of the request (for more information, see e.g., [7]).

For the sake of readability, a summary of the definitions appears in the last page of the chapter.

20.4 Expander Extraction

In this section we develop a simple, yet effective, technique for building a spander given an arbitrary expander. The first example we consider, namely, building a spander from the complete graph, is applicable to anonymous distributed networks where a centralized, deterministic solution cannot be used. Although the example is based on folklore results, it serves us in building intuition.

20.4.1 The Complete Graph

Consider the following generation of an expander: given a set of nodes, $|V| = n$, for each $v \in V$ choose d neighbors, independently at random. With overwhelming probability[1] the resulting graph is a good[2] *vertex* expander (for some constant $c > 0$):

$$P\left[\min_{S\subset V, |S|\le \frac{n}{2}} \frac{|\Gamma(S) \setminus S|}{|S|} < c\right] < o(1)$$

It is known that a vertex expander is also a good edge expander. Moreover, the average degree of such an expander is constant and the maximal degree is in $O(\log n/\log\log n)$ with very high probability.

To prove the above, we follow the proof in [33]. First fix a set $S \subset V$ of size s. Let $\Gamma(S) = \{u \in V | \exists v \in S, (u, v) \in E\}$ be the set of all neighbors of S. We wish to bound the probability that there exists a set T of size $t < c \cdot s$, such that

[1] We use the term "overwhelming probability" to denote a probability approaching 1 at least linearly with the size of the problem. For example, for a given n, $1-\frac{1}{n}$ is an overwhelming probability.

[2] An expander is considered "good" if it has a constant expansion parameter.

$\Gamma(S) \setminus S \subseteq T$. If no such set exists, then it holds that $\frac{|\Gamma(S) \setminus S|}{|S|} > c$. Furthermore, if there exists such T of size smaller than $c \cdot |S| - 2$, then there exists such T' of size exactly $c \cdot |S| - 1$. As a result, the probability that S is "bad" is

$$\begin{aligned}
\text{Prob}\big[S \text{ is bad}\big] &= P[\exists T : \Gamma(S) \setminus S \subseteq T] \\
&\leq \text{Prob}\big[\exists T : |T| = c \cdot |S| - 1 \wedge \Gamma(S) \setminus S \subseteq T] \\
&\leq \sum_{|T|=c\cdot|S|} P[\Gamma(S) \setminus S \subseteq T] \\
&= \binom{n}{c \cdot s} \cdot \left(\frac{s + c \cdot s}{n}\right)^{d\cdot s}
\end{aligned}$$

Using the above we show that the probability that the aforementioned construction has "bad" expansion is in $o(1)$.

$$\begin{aligned}
\text{Prob}\left[\min_{S\subset V, |S| \leq \frac{n}{2}} \frac{|\Gamma(S) \setminus S|}{|S|} < c\right] &= \text{Prob}\left[\exists S \subset V, |S| \leq \frac{n}{2} \text{ such that } \frac{|\Gamma(S) \setminus S|}{|S|} < c\right] \\
&\leq \sum_{s=1}^{n/2} \text{Prob}\left[\exists S \subset V, |S| = s \text{ such that } \frac{|\Gamma(S) \setminus S|}{|S|} < c\right] \\
&\leq \sum_{s=1}^{n/2} \binom{n}{s}\binom{n}{c \cdot s}\left(\frac{s + c \cdot s}{n}\right)^{d\cdot s} \\
&\leq \sum_{s=1}^{n/2} \left(\frac{n \cdot e}{s}\right)^{s}\left(\frac{n \cdot e}{c \cdot s}\right)^{c\cdot s}\left(\frac{s + c \cdot s}{n}\right)^{d\cdot s} \\
&= \sum_{s=1}^{n/2} \left[\left(\frac{s}{n}\right)^{d-c-1}\left(\frac{e}{c}\right)^{c} \cdot e \cdot (1+c)^{d}\right]^{s} \\
&\leq \sum_{s=1}^{\log n} \left[\left(\frac{\log n}{n}\right)^{d-c-1}\left(\frac{e}{c}\right)^{c} \cdot e \cdot (1+c)^{d}\right]^{s} + \\
&\quad \sum_{s=\log n}^{n/2} \left[\left(\frac{1}{2}\right)^{d-c-1}\left(\frac{e}{c}\right)^{c} \cdot e \cdot (1+c)^{d}\right]^{s} \\
&\leq \sum_{s=1}^{\log n} q^{s} + \sum_{s=\log n}^{n/2} r^{s} \quad \text{for } r, q << 1 \\
&\leq \frac{q}{1-q} + \frac{r}{1-r} = o(1)
\end{aligned}$$

If the initial graph is complete, a simple random choice of edges yields a good expander with high probability. Moreover, the average degree of such an expander is constant and the maximal degree is in $O(\log n / \log\log n)$ with very high probability.

To realize such a construction in a distributed manner, the need for a monitoring tool which validates the expansion of the spander arises. In the next section we will discuss such a self-organizing monitoring algorithm.

20.4.2 An Arbitrary Expander

In some cases, the initial topology of the graph is unknown at each node or cannot be stored at each node due to memory constraints (consider, for example, a peer-to-peer system with millions of nodes). The only input may be that the initial graph is a good expander. Yet, a spander construction needs to be carried out under such constraints; one may wish to employ the technique used for the complete graph to define a spander over *any* given expander graph. However, when the graph is not complete, choosing a constant number of neighbors at each node may result in non-expander graphs. As an example, consider the following graph: let $G = (V, E)$ be a regular (degree) expander graph. Now, let V_1 be a set of half the nodes in V and $V_2 = V \setminus V_1$. Consider the following graph, $G' = (V, E')$, where $E' = E \cup \{(v, u) : v, u \in V_1\} \cup \{(v, u) : v, u \in V_2\}$. Since adding edges can only increase the expansion of a graph, G' is a good expander (at least as good as G). Suppose we then proceed to generate a constant degree expander from G' by choosing for each node a constant number of neighbors, independently at random. It is easy to see that with a non-negligible probability, we will get a disconnected graph and, with an overwhelming probability, the resulting graph will not be a good expander.

Taking the above observation into account, we still wish to reduce the number of edges of an arbitrary expander without sacrificing the expansion property. When considering a graph $G = (V, E)$ such that each cut in the graph contains enough edges, implying that the graph is a good *edge* expander, one may notice that each edge is selected using a constant probability, with very high probability that all the cuts will remain large — thus keeping the *edge* expansion of the graph.

In the following Lemma we prove that starting from a graph with good enough edge expansion, namely with edge expansion in $\Theta(\log n)$ and a selection of each edge with a constant probability, results in an edge expander with overwhelming probability:

Lemma 1 *Let $G = (V, E)$ be a graph with edge expansion $c \cdot \log n$, where c is a constant and $n = |V|$. The graph $G^* = (V, E^*)$, such that $P[(u, v) \in E^*] = p$ has edge expansion of $pc \log n$ with overwhelming probability for appropriate p and c. For example, when $p = \frac{1}{2}$, $c = 48$ the probability is at least $1 - \frac{1}{n}$.*

Proof First consider a set $S \subset V$ such that $|S| = s \leq \frac{n}{2}$. Since G is an edge expander, we know that $h = |E(S, \overline{S})| > sc \log n$. Next, we calculate the probability that such a cut is "small" in G^*, applying Chernoff's inequality for the cumulative distribution function of the binomial distribution:

$$P\left[|E_{G^*}(S,\overline{S})| < \frac{ph}{2}\right] \le exp\left(-\frac{(ph - ph/2)^2}{2ph}\right)$$
$$= exp\left(-\frac{ph}{8}\right) \le exp\left(-\frac{psc\log n}{8}\right) = n^{-\frac{psc}{8}}$$

Now, using the union bound, and denoting a "bad" cut as a cut S such that $|E_{G^*}(S,\overline{S})| < \frac{ph}{2}$, the probability that no such "bad" cut exists is

$$P\,[!\exists \text{a "bad" set S}] \le \sum_{1\le s\le n/2} \binom{n}{s} n^{-\frac{psc}{8}} \le \sum_{1\le s\le n/2} n^{s\left(1-\frac{pc}{8}\right)} \le n^{2-\frac{pc}{8}}$$

It is easy to see that appropriate values of c, p (for example, $p = \frac{1}{2}, c = 48$) imply a probability of failure that is less than $1/n$.

Realizing such a construction in a distributed manner is simple; in order for each edge to be chosen with probability p, each node must choose each of its adjacent edges with probability $1 - \sqrt{1-p}$ (hence, the probability the edge is not chosen is $\sqrt{1-p}\cdot\sqrt{1-p} = 1 - p$). The edge is chosen if at least one node choses it.

20.5 Expansion Monitoring

Assuming that with high probability we can construct an expander, sometimes it might be necessary to evaluate our construction; we may wish to check that the resulting graph is a good enough expander (or even whether it forms a connected component) and whether enough edges were removed. Moreover, if the construction resulted in a good expander, we wish to preserve the current state of the network, so as to not disrupt service. As a result, periodically reconstructing the network is unacceptable, which implies the need for an algorithm to monitor the resulting construction.

When message sizes, memory, and processing power of a single node are not restricted, it is easy to collect the entire topology of the graph at each node and check if the resulting graph is a good expander. When restricting message sizes, memory requirements at each node, and convergence time to $O(\log n)$, such solutions are no longer feasible.

In the remainder of this section we tackle the task of monitoring the result of our construction when message sizes are limited to $O(\log n)$ by employing the mixing rate of expanders. We present a distributed expansion evaluation algorithm, which displays an inherent two-sided error; when the evaluated graph has expansion in

$\Theta(\log\ n)$, the algorithm returns "good" with overwhelming probability. When the graph has expansion less than $O(1/\log^3\ n)$, the algorithm returns "bad" with probability at least $\frac{1}{2}$. When the graph's expansion is in between the above values, our monitoring algorithm will return an arbitrary answer.

20.5.1 Monitoring by Random Sampling

Our first approach to monitoring is presented for the sake of building intuition, as the time it takes for detection is exponential. The monitoring is done by sampling the sets of nodes from the graph and calculating the expansion for each such set. If a set of nodes is found to be of small expansion, a reset procedure will follow which will reinitialize the nodes to a predetermined, consistent state.

Sets are sampled in the following manner: a node repeatedly starts a randomized propagation of information with feedback (PIF) flooding of the graph, which defines the set of nodes. Each of these sampled nodes will report back to the number of neighboring nodes which were not selected. This information will then propagate back to the initiating node which will then calculate the expansion of the selected set.

The main observation is given in Lemma 2.

Lemma 2 *Let $G = (V, E)$ be a graph. If there exists a set of nodes, $S \subset V$, $|S| \leq \frac{n}{2}$, such that $\frac{|E(S,\overline{S})|}{|S|} \leq c$ for some c, then there exists a subset $S' \subseteq S$, $|S| \leq \frac{n}{2}$, $\frac{|E(S',\overline{S'})|}{|S'|} \leq c$ and S' induces a connected subgraph of G.*

Proof The set S can be decomposed in the following fashion: $S = \bigcup_i S_i$, $\forall i, j : E(S_i, S_j) = \emptyset$ and $\forall i : S_i$ induces a connected subgraph, where each S_i is maximal (vertex-wise). Such a decomposition is easily obtained using greedy selection. Assume to the contrary that none of these subsets satisfies Lemma 2. It follows that for each subset S_i, $\frac{|E(S_i,\overline{S_i})|}{|S_i|} > c$. Now

$$\frac{|E(S,\overline{S})|}{|S|} = \frac{\sum_i |E(S_i,\overline{S_i})|}{\sum_i |S_i|} \leq c$$

$$c < \frac{\sum_i c \cdot |S_i|}{\sum_i |S_i|} \leq c$$

it follows that $c < c$, which is a contradiction. Hence, one of the sets S_i must satisfy Lemma 2. ⊠

More details on the way to implement analogous self-stabilizing monitoring will be presented in the next section as part of the description of the efficient monitoring scheme.

20.5.2 Mixing Rate-Based Monitoring

Expander graphs are rapidly mixing graphs ($O(\log n)$ mixing rate, implying that a random walk on the graph converges to the uniform distribution following $O(\log n)$ steps), and it follows that the cover time of such graphs is also short ($O(n \log n)$). We can employ this fact in the following way: assume that a node, v, wishes to check if the graph is rapidly mixing. v will start a random walk of length $O(n \log n)$ and attach a random color to this walk, chosen from a large enough domain. Furthermore, v associates three counters of $O(\log n)$ bits each to the random walk; one to limit the number of hops taken by the token, one to count the nodes discovered by the walk, and one to count the edges of the graph. Each time the walk visits a node for the first time (Fig. 20.1, line 1), the node increments the node counter by 1 and the edges counter by the number of its neighbors. Afterward, the token is transferred to a random neighbor (line 8). When the walk terminates (line 6), the counters are examined, either by the last node or by v after routing a message with the counters back to v. In case the walk covered less than n nodes, which implies that the graph is not rapidly mixing, or there are too many edges in the graph (relative to the original graph, which implies that the construction is not as productive as desired) a reset is initiated.

When the exact n is unknown, the protocol needs to be slightly adjusted; since counting the nodes cannot be used to determine coverage, each node should remember the last color of a token traversing it. After the random walk is terminated at node v, v initiates a flooding of the network to check if all nodes were colored by the same color of the token. If the flooding detects a node which has not been colored, a reset procedure will ensue.

```
$N$ = list of neighbors
$L$ = an upper limit on the length of the walk
$last\_seen = \emptyset$

**Receive**$(color, node\_counter, edge\_counter, length)$:
1   **if** $color \notin last\_seen$ **then**
2       $last\_seen \leftarrow last\_seen \cup \{color\}$
3       $node\_counter \leftarrow node\_counter + 1$
4       $edge\_counter \leftarrow edge\_counter + |N|$
5   **fi**
6   **if** $length > L$ **then**
7       **Report** counters to initiating node
8       **else**
9       choose $u \in N$ uniformly at random
10      **Send**$(u, color, counter, length + 1)$
11  **fi**
```

Fig. 20.1 Rules for node u

To speed up the detection rate we need to use a shorter random walk. To achieve this, v may send out many random walks in parallel instead of one, each of polylogarithmic length, and attach the same color to all of these walks. This elegant technique was suggested in [3] and analyzed for *regular expanders* (expanders with a regular edge degree); the proof therein is not applicable to non-regular graphs, as the authors employ the symmetric nature of the random walk on a regular graph in their proof. Namely, the probability that a random walk of length i visits a specific node at step i does not deviate by more than $\left(\frac{\lambda}{d}^{i}\right)$ (where λ is the second largest eigenvalue of the transition matrix and d is the degree of the graph) from the uniform distribution. Here, we employ more general bounds on the mixing time, derived from bounds on the stationary distribution of a random walk on an edge expander, to *extend* the analogy for the non-regular case.

Each of the random walks will hold counters in a fashion similar to the single random walk solution (see Fig. 20.1). These random walks will cover the graph with high probability (see Lemmas 4 and 5). To count the number of visited nodes, the final counters must be routed back to v (we will elaborate more on that in Sect. 20.5.3.4). If the total number of nodes visited is less than n then, with very high probability, the graph is not a good expander. Analogously to the discussion above, coloring and checking nodes can be used for the case in which n is not known.

In case the graph is not a good expander, we argue that initiating the random walks from a random edge results in failure with probability larger than half (see Lemma 6).

The proof determines, for a given walk length, the probability of visiting a specific node. Next, we calculate the number of walks needed to cover the entire graph with high probability. We begin by deriving a lower bound for the second largest eigenvalue of the Laplacian of the graph, which we later employ.

Lemma 3 *Let G be a connected, non-bipartite graph, such that the second largest eigenvalue of the Laplacian of G is λ and the expansion of the graph is h. Let $\Delta^2 = \frac{d_{\max}}{d_{\min}}$ be the ratio between the maximal and minimal degrees of nodes in G. Then $1 - \lambda > \frac{h^2}{2\Delta^8 d_{\max}^2}$.*

Proof Let $\pi(v)$ be the stationary distribution of v. It easily follows that for each v, $\frac{1}{2n\Delta^2} \leq \pi(v) \leq \frac{\Delta^2}{2n}$. Let $\Phi(S) = \Phi(\overline{S}) = \frac{|E(S,\overline{S})|}{2|E|\pi(S)\pi(\overline{S})}$ the conductance of a set $S \subset V$. The conductance of the graph is defined by $\Phi = \min_S \Phi(S)$. From [32], we know that $1 - \lambda > \frac{\Phi^2}{8}$. For each set S, such that $|S| < n/2$, we use the following:

$$2|E|\pi(S)\pi(\overline{S}) < 2|E||S||\overline{S}|\frac{\Delta^4}{4n^2} < \frac{\Delta^4 |S| d_{\max}}{2}$$

It follows that $\forall S$, $|S| < n/2$, $\Phi(S) \geq \frac{2|E(S,\overline{S})|}{\Delta^4 |S| d_{\max}}$. which further implies that $\Phi \geq \frac{2h}{\Delta^4 d_{\max}}$. Plugging into the bound given in [32], we get the desired bound. ☒

Lemma 4 *Using the notations of Lemma 3, let* $s = \frac{8\log n}{1-\lambda} > \frac{\log 4n\Delta^3}{\log\frac{1}{\lambda}}$. *The probability of a random walk of length* $2 \cdot s$, *starting at any vertex* $u \in V$, *to visit a given vertex* $v \in V$, *is at least* $\frac{\log n}{2\Delta^4 n}$

Proof We will follow the proofs presented in [3]. Fix u as the node from which the random walk starts and fix a node, v. Let Y_i, $s \leq i \leq 2s$, be the indicator of random variables such that $Y_i = 1$ if the walk visited v at step i. Let $Y = \sum_{i=s}^{2s} Y_i$ be the sum of these random variables. We will show that $P[Y > 0] > \frac{s}{4\Delta^2 n + 4s\Delta^4 + \frac{8\Delta^3 n}{1-\lambda}}$

Using the Cauchy–Schwartz inequality, we can see that

$$\left(\sum_{j>0} P[Y=j]\right)\left(\sum_{j>0} j^2 P[Y=j]\right) \geq \left(\sum_{j>0} jP[Y=j]\right)^2 \tag{20.1}$$

$$P[Y>0] = \left(\sum_{j>0} P[Y=j]\right) \geq \frac{\left(\sum_{j>0} jP[Y=j]\right)^2}{\left(\sum_{j>0} j^2 P[Y=j]\right)} = \frac{(E(Y))^2}{E(Y^2)} \tag{20.2}$$

From now on, we will focus on estimating both $E(Y)$ and $E(Y^2)$. From linearity of expectation, $E(Y) = \sum_{i=s}^{2s} E(Y_i)$ and for each i, $E(Y_i)$ equals the probability that the walk which started at u visits v precisely at step i.

In [32], the following bound is given, where $P^k(u,v)$ is the probability of a random walk of length k, started at u, to terminate at v:

$$|P^k(u,v) - \pi(v)| \leq \lambda^k \Delta$$

which implies the following:

$$\pi(v) - \lambda^k \Delta \leq P^k(u,v) \leq \pi(v) + \lambda^k \Delta$$

A simple calculation shows that when $s = \frac{8\log n}{1-\lambda} > \frac{\log 4n\Delta^3}{\log\frac{1}{\lambda}}$, for each Y_i and $k > s$ we get that $P(Y_i = 1) = P^k(u,v) \geq \pi(v)/2 \geq \frac{1}{4\Delta^2 n}$. From linearity of expectation, we get $E(Y) \geq \frac{s}{4\Delta^2 n}$.

We now turn our attention to calculating an upper bound for $E(Y^2)$. From the definition of Y we get

$$E(Y^2) = E\left(\left(\sum_{i=s+1}^{2s} Y_i\right)^2\right) = \sum_{i,j} E(Y_i Y_j) = \sum_i E\left(Y_i^2\right) + \sum_{i\neq j} E(Y_i Y_j)$$
$$= \sum_i E(Y_i) + 2 \sum_{s<i<j<2s} E(Y_i Y_j)$$

Now, $E(Y_i Y_j)$ is exactly the probability that the walk visits v at step i and then at step j $(i < j)$, which is the probability of visiting v at step i times the probability of returning to v after a walk of length $j - i$. The probability of returning to v after k steps equals $P^k(v, v) < \pi(v) + \Delta\lambda^k < \frac{2\Delta^2}{n} + \Delta\lambda^k$.

$$E(Y^2) = E(Y) + 2 \sum_{s\leq i<j\leq 2s} E(Y_i) P(Y_j|Y_i)$$
$$\leq E(Y) + 2 \sum_{s\leq i\leq 2s} E(Y_i)\left(\frac{s\Delta^2}{2n} + \sum_{k>0} \Delta\lambda^k\right) \leq E(Y)\left(1 + \frac{s\Delta^2}{n} + \frac{2\Delta}{1-\lambda}\right)$$

Therefore,

$$P[Y > 0] \geq \frac{E(Y)^2}{E(Y^2)} \geq \frac{E(Y)}{1 + \frac{s\Delta^2}{n} + \frac{2\Delta}{1-\lambda}} \geq \frac{\frac{s}{4\Delta^2 n}}{1 + \frac{s\Delta^2}{n} + \frac{2\Delta}{1-\lambda}} \geq \frac{s}{4\Delta^2 n + 4s\Delta^4 + \frac{8\Delta^3 n}{1-\lambda}}$$

Using the facts that $s = \frac{8\log n}{1-\lambda}$, $\log 1/\lambda > 1 - \lambda$ and plugging into the equation above, one can show that $P[Y > 0] > \frac{\log n}{2\Delta^4 n}$. ⊠

Lemma 5 *Using the notations of Lemma 3, $k = 4n\Delta^4$ random walks started from the same node in the graph, u, each of length $2s$, cover the entire graph with a probability of at least $1 - \frac{1}{n}$.*

Proof For each node v, the probability that it is not visited by a specific random walk is less than $1 - \frac{\log n}{2\Delta^4 n}$. The probability that none of the k random walks visit v is less than $\left(1 - \frac{\log n}{2\Delta^4 n}\right)^k$. When $k = 4n\Delta^2$, we get that this probability is smaller than $\frac{1}{n^2}$. Using the union bound, we get that the probability that all nodes are covered is larger than $1 - \frac{1}{n}$. ⊠

Corollary 1 *When $\Delta^2 < \log n$ and $h > \log n$, we get that $k < 4n \log^3 n$ random walks, each of length $2s$, where $s = \frac{8 \log n}{1-\lambda} < \frac{16\Delta^8 d_{\max}^2 \log n}{h^2} < 16 d_{\max}^2 \log^2 n \in O(\log^4 n)$, cover the graph with overwhelming probability.*

Corollary 1, in fact, implies that a single instance of a monitoring session on such a graph takes $O(\log^4 n)$ rounds and requires $O(n \log^7 n)$ messages.

Following lemma 5, we now know that if a graph is a good expander, the short random walks we use will cover the graph with high probability. However, we also wish to investigate the case in which a graph is not a good expander. The following Lemma illustrates that when a graph has less than constant expansion, the short random walks used will not cover the graph with a probability of at least half. For brevity, constants are omitted, and we assume that $\Delta \in o(\log n)$.

Lemma 6 *Let $G = (V, E)$ be a graph, such that there exists $S \subset V$, $|S| \leq \frac{|V|}{2} = \frac{n}{2}$ for which $|E(S, \overline{S})| < \frac{|S|}{8 \log^3 n}$. $n \log n$ random walks started from a uniformly chosen edge, each of length $O(\log n)$, will not cover the entire graph with a probability of at least $\frac{1}{2}$.*

Proof Let $(u, v) \in E$ be a directed edge, chosen uniformly at random. Start the $n \log n$ random walks from v. For each edge $e \in E$ define X_e as a random variable counting the number of times one of the random walks traverses e. Now, since (u, v) is chosen uniformly, which is the stationary distribution of the edges of the graph, for each directed edge $e \in E$ we have $E(X_e) = \frac{n \log^2 n}{|E|}$. Let $Y = \sum_{e \in E(S, \overline{S})} X_e$ (where e is a directed edge). Linearity of expectation implies

$$E(Y) = \frac{2n \log^2 n \cdot |E(S, \overline{S})|}{|E|} \leq \frac{2n \log^2 n |S|}{8n \log^3 n} = \frac{|S|}{4 \log n}$$

From Markov's inequality, we get that with a probability of at least $\frac{1}{2}$, the cut between S and $\overline{S}$ is not crossed more than $\frac{|S|}{2 \log n}$ times (in both directions). Assuming $v \notin S$, and since each walk is of length $\log n$, we cover at most $\frac{|S|}{2}$ nodes within S. If $v \in S$, we get that we cover even less of the nodes of $\overline{S}$, since $|S| \leq |\overline{S}|$. It follows that with a probability of at least $\frac{1}{2}$, we do not cover the entire graph.

20.5.3 Self-Stabilizing Distributed Monitoring

In the next sections we present a self-stabilizing distributed monitoring algorithm to monitor the mixing rate of a graph. The algorithm is based on the repeated selection of a random edge from the graph and starting random walks from this edge. Whenever a problem is observed in the graph, the snap-stabilizing reset that we describe here will be utilized to reset the monitoring algorithm and to rebuild the current graph (if it is not a good spander). As the entire construction and monitoring must be logarithmic in the number of nodes, we had to introduce, reform, and augment new techniques.

20.5.3.1 Snap-Stabilizing Data Link

A distributed algorithm is termed *snap-stabilizing* if the algorithm stabilizes following the first request by any node and before, or simultaneously with, a notification arriving to the requesting node at the completion of the request.

Throughout the text we assume the use of a snap-stabilizing data-link layer. We consider several algorithms which can be used to realize a snap-stabilizing data-link layer.

- **Spontaneous receiver**. When the receiver may send duplicated answers for a single frame sent (or the duplication may be caused by the underlying physical layer), we suggest the following algorithm (see [24]): when the sender, s, wishes to send a frame, f, to the receiver, r, s will send f to r repeatedly and attach a sequence number to f. At first, s will attach the number 1 to f and repeatedly send f to r with sequence number 1. Once s receives an acknowledgment from r upon the receipt of f with sequence number 1, s will repeatedly send f with sequence number 2. s will keep incrementing the sequence number of f each time s receives an acknowledgment on the current sequence number, until s reaches $2 \cdot lc + 1$ times (where lc is the bound on the link capacity). At this point f is assured to be delivered at r.
  When r receives a frame f with a sequence number $2 \cdot lc + 1$, following a frame with a different sequence number, r will deliver the frame upward in r's network stack.
- **Non-spontaneous receiver**. When frames are not duplicated (either by r or by the physical layer), we suggest the following algorithm: s will first repeatedly send f to r, but will mark f with 0. s will then count the number of acknowledgments it receives from r. When s has received $2 \cdot lc + 1$ acknowledgments, s will then mark f with 1 and repeat the process. After s receives $2 \cdot lc + 1$ additional acknowledgments for f, f is assured to be delivered at r.
  When r receives a frame f with a mark 0 that is immediately followed by a frame with a mark 1, r will deliver the frame upward in r's network protocol stack.
  The algorithm for the non-spontaneous receiver is more efficient both communication-wise and time-wise.

20.5.3.2 Snap-Stabilizing Message Passing Reset

A reset procedure ensures that once it is initiated, and before the reset is terminated, each node receives a logical "reset" signal (possibly resetting the node's state to a predetermined state). Moreover, following a logical reset at node u, if u sends a message to node v, then node v will have received a reset signal before processing the message from u.

We present a reset procedure for message passing networks. The reset is executed on a graph G, which in our case is of diameter $d \in O(\log n)$. We assume the use of a snap-stabilizing data-link protocol for passing messages between nodes. A similar technique appears in [4, 5, 10], although here we present and prove *snap-stabilization and termination*, while fitting anonymous networks. The reset procedure for a single node, u, appears in Fig. 20.2.

The reset procedure is based on bounded counters of $3d$ at each node. Each node u maintains a reset counter. To initiate a reset, u just sets u's counter to zero (line 1). The technique used by the reset procedure is that a node, u, will only increment u's counter after u is certain that all of u's neighbors have counters which are greater than or equal to u's counter. This is achieved by first saving the counter values received from the neighbors (line 3) and assigning u's counter with the minimal value among the counters of all of u's neighbors, plus one (line 14).

u must also repeatedly broadcast u's counter value to u's neighbors. The broadcast is based on the snap-stabilizing data-link algorithm presented above. The SnapSend procedure used in line 17 ensures that the counter value C is sent to the

```
$cnt_u$ = a counter
$d$ = the diameter of the graph
$N[i]$ = the last counter received
          from neighbor $i$
$flag$ = a boolean flag

**Reset**
1   $cnt_u \leftarrow 0$
2   $flag \leftarrow true$

**Receive** Counter value $c$ from neighbor $v$
3   $N[v] \leftarrow c$
4   **if** $c < cnt_u$ **then**
5       $flag \leftarrow true$
6       $cnt_u \leftarrow c + 1$
7   **fi**

**While** $cnt_u < 3d$
8   **while** $flag = true$ **do**
9       $flag = false$
10      **Broadcast**()
11  **done**
12  **if** $\forall i \in N : N[i] \leq cnt_u + 1$ **then**
13      $flag \leftarrow true$
14      $cnt_u \leftarrow \min\{cnt_u, \min_i N[i]\} + 1$
15  **fi**

**Broadcast**
16  **foreach** neighbor $v$ **do**
17      $c \leftarrow$ **SnapSend** $cnt_u$ to $v$
18      **Receive**$(c, v)$
19  **done**
```

Fig. 20.2 Snap-stabilizing reset procedure for u

neighbor v using a snap-stabilizing data-link algorithm, thus ensuring the delivery of the new value. The receiving node of the SnapSend procedure piggybacks its own counter value while sending acknowledgments, which is returned from the SnapSend procedure on the initiator side. This ensures that upon the termination of a single SnapSend procedure both ends hold the other side's counter values that existed during the SnapSend execution.

The $flag$ variable is used to ensure that u will not reassign u's counter before updating all of u's neighbors with u's current value. This property is vital to our proof and establishes a happened before relation between counter updates across the graph.

The proof is based on the following observation: when a node, v, assigns 0 to its counter (starting a reset), this value will propagate in the graph, causing other nodes to adopt a counter value not greater than their distance to v. First, v's immediate neighbors will set their counters to at most 1. Afterward, v's neighbors' neighbors will set their counter to (at most) 2, and so on. We then show that when v reaches $2d$, the counter of each node cannot exceed $3d$.

Definition 1 Given an execution $\mathcal{E}$, a *happened before path* during $\mathcal{E}$ between a node u and a node v is a path in the graph induced by a happened before relation between u and v. Assume the happened before relation is $u \rightsquigarrow p_1 \rightsquigarrow p_2 \rightsquigarrow \cdots \rightsquigarrow p_i \rightsquigarrow v$, where u sent a message to p_1, p_1 sent a message to p_2 after receiving the message from u, etc. The *happened before path* is defined to be $u, p_1, p_2, \ldots, p_i, v$.

Lemma 7 *Let c_v be a configuration in which a node, v, has started the reset (line 1). For every node u, and in every fair execution, there exists a configuration c' such that u receives a counter value (line 2) in c' and a happened before relation between the atomic step of v in c_v, which started the reset, and the atomic step of u in c' exists. Moreover, there exists such a happened before relation and a configuration c_u such that the happened before path induced by the happened before relation forms a shortest path from v to u.*

Proof The proof is by induction over the distance between u and v. When $d(u, v) = 0$, the claim obviously holds as $u = v$. Assume that the claim holds for all $p \in V$ such that $d(p, v) < d(u, v)$. Since the graph is connected, there exists a node p such that $d(u, v) > d(p, v)$ and $(u, p) \in E$. According to the induction assumption, there exists a configuration c_p satisfying the lemma. Let c_u be the first configuration following c_p in which u receives a counter value from p which was sent in a configuration following c_p. Clearly, c_u satisfies the conditions of the lemma. $\boxtimes$

Lemma 8 *Let c_v be a configuration in which a node, v, has started the reset (line 1). For a node u, let c_u be the earliest configuration satisfying Lemma 7. The counter value of u in c_u, $cnt_u(c_u)$, may not exceed $d(v, u)$, the distance between v and u.*

Proof By induction on $d(v, u)$. When $d(u, v) = 0$, the assumption clearly follows since $v = u$ and $c_u = c_v$. Assume that the claim holds for all values smaller than i. Let u be a node such that $d(v, u) = i$ and mark c_u as the first configuration which satisfies Lemma 7. Let w be u's neighbor such that $d(w, v) = i - 1$ and

w is u's predecessor in the happened before relation which defined c_u. Let c_w be the configuration which satisfies Lemma 7 with regard to w. It follows, from the inductive assumption, that $cnt_w(c_w) \leq i - 1$. Since w repeatedly broadcasts its counter value and will only increase w's counter value after the broadcast phase is completed at least once (this is due to the $flag$ variable), there exists a configuration c', between c_w and c_u, in which u received a value $i - 1$ or less from w for the first time. It follows that $c' = c_u$, and since u adopts the counter value (plus one) received if the value is smaller than its own, we conclude that u's counter value may not exceed $(i - 1) + 1 = i$ in c_u. ⊠

Lemma 9 *Let c_v be a configuration in which a node, v, has started the reset (line 1). For a node, u, let c_u be the earliest configuration satisfying Lemma 7, and let w be a node on the happened before path from v to u. For every fair execution, and in every configuration c' between c_w and c_u, $cnt_w(c') \leq d(u, v) + d(u, w)$.*

Proof Let u be a node and c_u be a configuration as defined in Lemma 7. Let w be a node on the happened before path from v to u. The proof is by induction on $d(u, w)$. For $d(u, w) = 0$, it follows that $u = w$ and the claim clearly holds from Lemma 8. Assume the claim is correct for all values smaller than i.

Let x be a node on the happened before path between u and w, such that $d(u, x) + 1 = d(u, w)$. First note that according to Lemma 8, we know that $cnt_w(c_w) \leq d(w, v)$. From the induction assumption, it follows that in every configuration c' between c_x and c_u $cnt_x(c') \leq d(u, v) + d(u, x)$. Furthermore, in each such c' it clearly follows that $cnt_w(c') \leq cnt_x(c') + 1$, since w always adopts the lowest counter value in its neighborhood plus one. As a result, in every configuration c' between c_x and c_u, we get that $cnt_w(c') \leq d(u, v)+d(u, x)+1 = d(u, v)+d(u, w)$.

According to Lemma 8, $cnt_w(c_w) \leq d(w, v)$. Note now that w will only increment its counter in consecutive steps and will make sure that w communicates with all of w's neighbors afterward; this is due to the $flag$ used in the algorithm. This in turn ensures that between c_w and c_x w may increment w's counter by not more than 1. Hence, in every configuration c' between c_w and c_x we get that $cnt_w(c') \leq d(w, v) + 1$.

To conclude, we get that in every configuration c' between c_w and c_u, $cnt_w(c') \leq d(u, v) + d(u, w)$. ⊠

Lemma 10 *Let c_v be a configuration in which a node, v, has started the reset (line 1) and u be a node such that for each node w c_w preceded c_u. It follows that in every configuration c' between c_w and c_u, $cnt_w(c') \leq 3d$.*

Proof Assume, toward contradiction, that there exists a node w and a configuration c' such that c' is between c_w and c_u and $cnt_w(c') > 3d$. Consider the shortest path between v and w, $(v, v_1, v_2, \ldots, v_k, w)$ induced by Lemma 7. It follows that $cnt_{v_k}(c') > 3d - 1$, since w and v_k have exchanged counter values. In a similar way, we can show that $cnt_v(c') > 3d - i - 1$. Now, since $i \leq d - 1$, it follows that $cnt_v(c') > 2d$. However, according to Lemma 9, we know that in each configuration between c_v and c_u, the counter value of v may not exceed $2d$, which yields the contradiction. ⊠

Lemma 11 *Let c_v be a configuration in which a node, v, started a reset. In every fair execution there exists a configuration c', which follows c_v after at most d communication rounds, such that for each node u it holds that $cnt_u(c') < 3d$.*

Proof The proof follows from the fact that after d communication rounds there is a happened before relation between v and any nodes u on a shortest path between v and u. Moreover, from Lemma 10 it follows that all nodes have counter values less than $3d$.

Lemma 12 *Liveness: let c be a configuration in which at least one node has a counter value less than $3d$. Let v be a node such that $cnt_v(c) < 3d$ and for every node u $cnt_u(c) \geq cnt_v(c)$. Let c' be a configuration following c after d communication rounds, during which no node has started a reset. For every node $u \in V$, $cnt_u(c') \geq cnt_v(c) + 1$.*

Proof The proof is by the fact that every node v with a minimal counter will not set v's $flag$ to $true$ and hence will finish the broadcast phase and advance v's counter value by 1.

To ensure the termination of the reset algorithm, we enable nodes to start a reset only when their counter value is larger than $3d$. Such a restriction gives rise to the following: assume a node, v, has started the reset. Following Lemma 11, we will reach a configuration in which all counter values are less than $3d$. Combining that with Lemma 12 and assuming that once a successful reset is performed, no new reset will be started for a long time, such that nodes reaching $3d$ will not start a reset for a long enough period, the reset algorithm will terminate. Namely, all nodes' counters will reach $3d$.

Theorem 1 *The reset protocol of Fig. 20.2 is a snap-stabilizing reset protocol, which terminates after at most $3d$ rounds following initialization.*

The snap-stabilization paradigm requires that once one node initiates the (reset) algorithm, the algorithm will terminate successfully, namely, the initiator receives a notification on the termination and no more messages are sent. The termination property of the snap-stabilizing reset is a rare property in the scope of self-stabilizing algorithms (see [2]). The reset algorithm is snap-stabilizing since once a node, v, starts the reset algorithm, each node in the system will go through a reset, namely lower its counter value below $3d$, before v receives an indication on the reset termination.

In the context of our monitoring algorithm presented in the following section, a leader must be elected following the reset, and a BFS tree rooted in the leader needs to be defined. We suggest the following algorithm: once the counter of a node, v, reaches $3d$, v will start broadcasting to v's neighbors v's candidate for the BFS leader and the distance to the candidate, starting with v: $(v, 0)$. v will repeatedly collect the BFS leaders of v's neighbors. If there exists a BFS leader with a lower identifier than v's current BFS leader or one of v's neighbors is closer to the leader than v, v will adopt this BFS leader, mark the node from which v received this BFS leader as v's predecessor in the BFS tree, and add one to the distance to the BFS leader. Once any BFS leader has a distance larger than d, v can safely discard this BFS leader.

It is easy to see that this leader election algorithm, when started after a reset, will elect a leader, define a BFS tree rooted in the leader, and establish correct distances. After further $2d$ communication rounds (due to the liveness property, see Lemma 12) all nodes will agree on one leader, the node with the lowest id in the system. If we let nodes continue the counting of the reset algorithm, starting to execute the leader election algorithm at $3d$ and up to $5d$, then the leader can start a new monitoring session once its counter reaches $5d$. In the sequel, once the counter reaches $5d$, the tree structure remains fixed in the sense that parent pointers are not changed.

20.5.3.3 Monitoring Algorithm for a Single Node

We next present a self-stabilizing expansion monitoring algorithm. In fact, given the self-stabilizing reset presented earlier, a simpler non-stabilizing version can be used, as long as there is a local predicate for checking the consistency of the monitoring that triggers the reset. Still, the self-stabilizing monitoring may be of independent interest. Next, we consider an instance of the algorithm which is started from one node, which we denote ml (for monitoring leader). The monitoring algorithm is a self-stabilizing version of the technique presented in Sect. 20.5. Namely, a node starting a monitoring session will send the required number of tokens (see Lemma 5), each performing a random walk, counting the number of nodes and edges in the graph.

When designing the self-stabilizing version of the monitoring algorithm, we may notice two problems which we need to solve; one, we need to ensure that tokens can be routed back to ml once they have traveled the required length. Second, if ml assumes that a monitoring session has started, and none of ml's tokens exist in the graph, we need to prevent ml from waiting forever. To overcome both of these obstacles, we use the same repeated token sending mechanism.

Each node u will follow the algorithm presented in Fig. 20.3: ml will add a serial number to each token it sends, e.g., they will be consecutively numbered, $t_1, t_2, \ldots$. Each node u records not only the color of the token, but also, for each token t_i, to which neighbor, w, u sent token t_i following the last arrival of t_i in u (lines 8 and 11). We call this the *forward pointer* of t_i at u. u then repeatedly sends all the tokens u has received to ensure delivery (line 20). We use the repeated send technique to argue concerning the termination of each monitoring phase.

To ensure the delivery of the tokens back to ml, when they have reached the end of their random walk, we need the following mechanism: when a random walk terminates (reached its maximum hop count), the node in which the walk terminated saves the information collected by the walk (line 11). Each time a node u receives a token (whether for the first time or not), if u currently holds an answer to the token, u forwards the answer to the sender of the token (line 15).

We next show that the traversal of a single token sent by ml will terminate, and the information contained in the token (node count and edge count) will be propagated back to ml. The proof considers one token, t_i. Denote by k the length of the random walks according to the previous section ($k \in O(\log n)$).

```
$N$ = list of neighbors
$L$ = an upper limit on the length of a walk
$tokens[]$ = an array, holding the last tokens received
$answers[]$ = an array, holding the last hop of a token (if exists)

**Start Walks**
choose a random color $c$
**for** $i \in tokens$ **do**
    choose a random neighbor $p$
    $tokens[i] \leftarrow (c, 0, 0, 0, p)$
**done**

**Receive**$(color, index, origin, hop\_counter, node\_counter, edge\_counter)$ from $v$
choose a random neighbor $p$
**if** $tokens[index].color \neq color$
    $tokens[index] \leftarrow (color, hop\_counter, node\_counter + 1,$
                        $edge\_counter + |N|, p)$
**else if** $tokens[index].hop\_counter < hop\_counter$ **then**
    $tokens[index] \leftarrow (color, hop\_counter, node\_counter,$
                        $edge\_counter, p)$
**fi**
**if** $answers[index].color = color$ **then**
    **Send** $(i, answers[index])$ to $v$
**fi**

**Repeatedly**
**foreach** $i \in tokens$ **do**
    $(color, hop\_counter, node\_counter, edge\_counter, p) \leftarrow tokens[i]$
    **if** $hop\_counter < L$ **then**
        **Send** $(color, i, v, hop\_counter + 1, node\_counter, edge\_counter)$ to $p$
    **else**
        $answers[i] \leftarrow (color, node\_counter, edge\_counter)$
    **fi**
**done**

**Receive**$(index, answer)$ from $v$
$(c, hop\_counter, node\_counter, edge\_counter, p) \leftarrow tokens[index]$
**if** $color = answer.color$ **then**
    $answers[index] = answer$
**fi**
```

Fig. 20.3 Self-stabilizing monitoring algorithm for u

Definition 2 We say that a node v contains a token t_i in configuration c if one of the following conditions holds:

- v has received t_i in c for the first time.
- v has a forward pointer for t_i (with a hop count of $j < k$), which points at u, and the maximal hop count of t_i in u (if it exists) is smaller than j.

Lemma 13 *Let c_0 be a configuration, and T_c be the set of all instances of t_i in a configuration c (t_i might appear twice due to faults). Further, let c_k be a configuration following c_0, in a fair execution, after the first k communication rounds. Assuming that ml has not initiated a new monitoring phase using a new color, then $T_{c_k} = \emptyset$.*

Proof Nodes repeatedly forward token messages to their neighbors. As a result, if at configuration c_i node v contained a token, then after one communication round v would have forwarded the token and increased the token's hop counter.

Define $H(T_c)$ as the minimal hop counter of a token in T_c and c_{i+1} as the first configuration following a single communication round. It immediately follows that $H(T_{c_i}) < H(T_{c_{i+1}})$. Now, since tokens are terminated after reaching a large enough hop counter, k, we get that after k rounds, starting in c_0, there can be no tokens in c_k. ⊠

It easily follows from Lemma 13 that for any configuration c_j, such that $j \geq k$, if *ml* does not initiate a new monitoring phase, no node will change its forward pointer that is related to t_i.

Lemma 14 *Under the notation of Lemma 13, in c_k there exists a path from ml to a node t, in which t_i terminated, which is defined by the forward pointers of t_i starting from ml. Moreover, this path will not change unless ml initiates a new monitoring phase.*

Proof Let c_m, $c_0 \leq c_m \leq c_k$, be the last configuration in which *ml* contained t_i (if no such m exists, set $m = 0$). Since *ml* does not hold t_i between c_m and c_k, *ml*'s forward pointer for t_i will not change.

Let v_1 be the node to which *ml*'s forward pointer points. Let c_{m_1} be the last configuration in which v_1 contained t_i. There exists such a configuration c_{m_1}, since according to Lemma 13 there are no tokens in c_k. Since v_1 does not hold t_i between c_{m_1} and c_k, v_1's forward pointer for t_i will not change.

Using the same argument, we define a sequence of nodes, $r = v_0, v_1, v_2, \ldots, v_l$, such that v_j's forward pointer for t_i points at v_{j+1} in c_k and will not change until *ml* starts a new monitoring phase.

Lemma 15 *Under the notation of Lemma 13, after further k communication rounds at most ml will receive a notification for the termination of t_i.*

Proof The proof is by induction, showing that after each round, the notification advances at least one node backward on the path defined by Lemma 14. ⊠

The next theorem follows immediately from the lemmas above. In particular it follows that after the first time *ml* initiates a new monitoring phase using a new color, the monitoring phase will complete successfully. Thus, assuming that *ml* initiates t random walks, each of length k, in a system with N nodes, we get the following:

Theorem 2 *The monitoring algorithm is a self-stabilizing algorithm which stabilizes in $O(k)$ rounds. Furthermore, the monitoring algorithm requires $O(t(\log t + \log k + \log N))$ bits of memory at each node.*

20.5.3.4 Global Monitoring Algorithm

Given the monitoring algorithm and the reset procedure above, we present a self-stabilizing expansion monitoring algorithm. The algorithm is based on repeatedly invoking the monitoring algorithm for a single node, each instance from a different node.

To coordinate between the nodes and to ensure only one monitoring session is active at a time, we employ a single leader in the system, with a BFS tree rooted in this same leader. Repeatedly, the BFS leader, *bl*, selects a directed edge, $\langle u, v\rangle$, uniformly among all directed edges, and informs v, the edge's endpoint, to start a new monitoring session as the monitoring session leader. If a specific percentage of monitoring sessions failed, *bl* may conclude that the graph is not a good expander and initiate a reset to rebuild the graph.

We next detail the specific techniques used to realize the global monitoring algorithm.

- **Self-Stabilizing BFS tree**. The BFS tree is assumed to be defined. Nodes will repeatedly inspect the status of the tree, and upon finding any error will initiate a reset. Namely, each node v repeatedly checks that v's neighbors have the same leader v has. In case v is the leader, v also checks whether its distance to the BFS root is 0. Otherwise, when v is not a leader, v checks whether v's parent in the BFS is closer to the leader than v by exactly one and there is no neighbor that is closer to the leader. Once any node (either a leader or a non-leader node) detects a violation of the above assertions, the node will initiate a snap-stabilizing reset, ensuring that a single BFS leader will be elected and a correct BFS tree will be constructed.

To facilitate communications between the BFS leader and the rest of the nodes, the BFS leader will repeatedly color the BFS defined by the parent child relation, using randomly selected colors. Such a self-stabilizing coloring technique using broadcast and convergecast over a tree has been well investigated, see [10].

When coloring the BFS tree with a new color, the BFS leader may piggyback messages on the messages used for coloring, and nodes may piggyback information back to the BFS leader with their replies.

- **Selecting the next monitoring node**. Each monitoring session must start from a directed edge, chosen uniformly. To select the edge, the BFS leader, *bl*, will employ the coloring of the tree defined above.

During and after the construction of the BFS tree each node v will be repeatedly notified by its children on se_u, the number of edges connected to all nodes in the subtree rooted in each child u. v will repeatedly define se_v to be $\Delta_v + \sum_{u \text{ child of } v} se_u$, where Δ_v is the number of edges directly connected to v. In particular, a leaf u will repeatedly notify its parent with Δ_u. Moreover, each node maintains a *local* ordering of the node's children.

Each node u associates a set of natural numbers between 1 and se_u with u's children and with u itself, so that each number is associated with a single node and each node v receives a set of numbers, N_v, of cardinality se_v so that $N_v = \{i_1, i_2, \ldots, i_{se_v}\}$.

The BFS leader, *bl*, initiates the selection of the new node by first selecting, uniformly at random, a natural number k between 1 and se_{bl}. If k is associated with *bl*, *bl* is the new monitoring leader. Otherwise, let v be *bl*'s child with which k is associated, and let $k = i_j \in N_v$. *bl* will inform v (by piggybacking on the coloring algorithm's messages) that v should continue the process with the number j.

When a node v receives a message that v should select the next monitoring leader using the number k, v first checks with which node k is associated. If k is associated with v itself, v selects itself as the new monitoring leader. Otherwise, assume k is associated with a child, u, such that $k = i_j \in N_u$. v will notify u that u should continue the process with the number j. Note that the selection algorithm always terminates and selects a node with a uniform distribution over all edges' endpoints.

At the end of the coloring of the BFS tree, *bl* learns the identity of the node which was selected as the next monitoring session. Moreover, the natural number, k, which *bl* has randomly chosen, uniquely identifies the next monitoring leader. *bl* will then initiate a new coloring of the tree, in which *bl* broadcasts the identity of the new monitoring leader. Next, we describe the mechanism used to ensure that the monitoring leader indeed performs the monitoring.

- **Ensuring a monitoring leader exists**. *bl* must ensure that the current monitoring leader is active. There are several ways to achieve this, amongst them is using the coloring to communicate with the current monitoring leader. A different way, which does not involve the coloring algorithm is by routing messages directly to the monitoring leader, down the BFS tree, by employing the same mechanism used to select the monitoring leader; as *bl* remembers k, the randomly chosen natural number which identifies the monitoring leader, *bl* can route messages to the monitoring leader which can then send replies up the tree toward *bl*. To ensure self-stabilization, each time *bl* sends a message, *bl* will attach a random color to the message and expect a reply containing the same random color.

If *bl* does not receive a confirmation that a monitoring leader exists, *bl* will perform a reset to reinitialize the BFS tree.

- **Detecting bad expansion**. There exists a small constant c, such that if the monitoring algorithm (for a single node) indicated that the graph is not a good expander at least $c/2$ times when running c monitoring sessions consecutively (from different nodes), then with probability $1 - o(1)$ the graph is not a good expander. This follows by examining the conditional probability that the graph is not a good expander given that the monitoring algorithm indicated so in more than half the times of c successive sessions, and that the probability of a wrong answer on a good expander and the probability of getting a good expander a priori are both $1 - o(1)$.

bl will employ this constant to check the last c invocations of the monitoring algorithm. If more than $c/2$ invocations failed, *bl* will initiate a reset to rebuild the graph. We wish to draw the reader's attention to the fact that monitoring results may not be reused; for example, once c successive monitoring sessions are finished, a new set of successive monitoring sessions will start.

Note that one may employ more than one monitoring session in parallel by uniformly choosing a set of edges which, in turn, define a set of nodes such that each such node is a monitoring leader, hence boosting the speed of the detection time.

- **Stretch factor**. The diameter of the resulting spander graph is in $O(\log n)$. This, in turn, implies that the spander has an additive stretch factor in $O(\log n)$.

20.6 Distributed Hierarchical Spanner Construction

Given a graph $G = (V, E)$ satisfying the conditions of Lemma 1 (namely, having an edge expansion of at least $c \cdot \log n$), we wish to define a hierarchical structure of spanders such that each spander will have fewer edges than the one before it (and, as a by-product, smaller expansion). This hierarchical construction can then be used to ensure the quality of service in a wide variety of applications; the system can automatically adjust the communication graph used in order to achieve its goals, taking into account the underlying structure of the chosen graph and the fitting number of edges versus the probability of expansion.

We propose a self-stabilizing and self-organizing distributed algorithm based on the techniques shown above. Namely, we propose to distributively define a sequence of spanders, $\{G_i\}_{i=0}^{k}$, $G_0 = G$, where k is an input to the construction algorithm, such that each graph G_i results from G_{i-1} by applying the algorithm portrayed in Sect. 20.4.2. Furthermore, to ensure that each spander is indeed up to the standard, we apply the monitoring technique portrayed in Sect. 20.5.2 by starting a monitoring phase from each node and for each spander in the hierarchy. If one node discovers that a given spander G_i is either not sparse enough or not a good expander, it will start a snap-stabilizing reset which restarts the construction from G_i onward.

It immediately follows that the proposed hierarchical construction is self-stabilizing; since each graph of the hierarchy is constantly monitored by a self-stabilizing monitoring protocol, once one of the graphs is found to be faulty the snap-stabilizing reset protocol will tear down the hierarchy from the faulty graph forward (using the reset procedure). The construction will then resume from the latest graph known to be a good expander. It follows, according to the fair composition technique ([10], Sect. 2.7), that the hierarchical construction will eventually stabilize.

- **Maximal hierarchy height**. When considering possible values for k, the height of the hierarchy, several considerations come to mind; assuming that the probability of edge selection in each level of the hierarchy is p, at each level i there are at most $p^i|E_0|$ edges. In fact, we check that the number of edges in each level is not too large. As a result, when $i \in O(\log n)$ the spander construction will fail. This implies that at most $O(\log n)$ levels in the hierarchy can exist, and therefore appropriate memory and communications resources for this number of levels are needed.
- **Self-organization**. The construction is self-organizing; the construction of the graph is a local computation carried by each node locally and independently of neighboring nodes. Each instance of the monitoring protocol takes $O(\log n)$ communication rounds to complete. Taking into account that following a constant number of successful monitoring phases, with very high probability the graph

is a good expander, we then get that a hierarchy of height k will stabilize in $O(k \log n)$ communication rounds, if a reset procedure is invoked at every level of the hierarchy in the worst case.

Another property of self-organizing algorithms is fast reaction time to topology changes after the algorithm has converged. Consider a node, v, joining G_0. If G_0 will remain a good expander and satisfy the conditions of Lemma 1, then the following simple procedure, carried by v, will ensure that each graph in the hierarchy will be a good expander: v will select each edge of G_i $(i > 0)$ to be in G_{i+1} with probability p (the same probability as in the construction algorithm); each G_i can still be seen as a random graph drawn from the probability space induced by our construction algorithm, from which it follows that each graph is a good expander with high probability.

Now, consider the effect v has on the monitoring algorithm. We first require that v will join the BFS tree by selecting the neighbor with the shortest distance to the BFS leader as v's BFS parent. v will then notify v's parent with Δ_v, such that the sum of degrees held by each node on the path from v to the BFS leader will also be updated. Next, v will check that the height of the BFS tree is not too big, say, bigger than $2 \log n$. If the height is too big, v will initiate a reset to rebuild the BFS tree.

Overall, if the pattern of nodes joining G_0 is symmetric, meaning each node joining the graph is connected to other nodes using a uniform distribution, then the number of nodes joining the network between two successive resets is large. Hence, the expected, in fact, amortized, convergence time following a join is short and in $O(1)$.

Next, consider a node, v, leaving the graph or crashing. If the basic graph, G_0, remains an expander with the required properties as in Lemma 1 even after v is removed, then from a similar argument as the one for nodes joining the network, each graph G_i in our hierarchy will remain a good expander with high probability. Let us now consider the implications of v's crash on the monitoring algorithm.

Each of v's neighbors in G_0 will detect that v has crashed. Each node u such that v was u's parent in the BFS tree will need to select a different parent and update this new parent with se_u. Moreover, v's parent, w, will need to deduce se_v from se_w. Regarding the current monitoring session, each node v will need to check if v is currently holding a token which it needs to forward to a crashed node. If so, v will need to forward the token to a different node.

If v was the monitoring leader, then a new monitoring leader will be selected by the BFS leader. In the case that v is itself the BFS leader, a new reset procedure will be started.

Overall, the correct behavior of the monitoring algorithm will not be harmed. The total communication round which it would take to update the BFS tree following a crash is expected to be in $O(1)$ when the probability of node failure is uniform.

Furthermore, the same argument holds when edges are removed or added, as long as G_0 satisfies Lemma 1. As a result, the construction algorithm converges in an expected (amortized) $O(1)$ communication rounds following a topology change.

- **Hierarchical reset**. Consider the reset protocol presented earlier, which we perform on the original graph, G, when building the hierarchy. The message complexity of the reset algorithm is in $O(d \cdot |E|)$, where d is the diameter of the graph and E is the set of edges. When G is sparse, the reset algorithm is efficient. However, when G is dense, i.e., $|E| \in O(n^2)$, the communication complexity of the reset algorithm is in $O(n^2)$.

To lower the communication cost of the reset algorithm we propose to perform the reset algorithm associated with the monitoring algorithm of G_{i+1} on G_i. However, one problem may arise; we need to ensure that G_i is indeed connected. Otherwise, the reset algorithm will fail to reset the entire graph. To ensure that G_i is connected, we augment the reset protocol in the following way: the BFS leader will count the nodes in the tree in addition to coloring the tree. If the number of nodes is too small or differs from the number of nodes in $G = G_0$, then a reset on G_{i-1} will need to be started to rebuild G_i. Alternatively, the BFS of G_0 will be used to update the leader identity of each level.

Theorem 3 *The proposed hierarchical distributed expander construction algorithm is self-stabilizing and self-organizing. The convergence time of a hierarchy of height k is in $O(k \log n)$ and the memory required at each node, imposed by the monitoring algorithm, is in $O(kn \log n)$.*

References

1. The virtual infrastructure project. `http://groups.csail.mit.edu/tds/vi-project`.
2. I. Abraham, and D. Malkhi. Probabilistic quorums for dynamic systems. *Distributed Computing 18*, 2:113–124, (2005).
3. N. Alon, C. Avin, M. Koucký, G. Kozma, Z. Lotker, and M. R. Tuttle. Many random walks are faster than one. In *SPAA 2008: Proceedings of the 20th Annual ACM Symposium on Parallel Algorithms and Architectures, Munich, Germany, June 14–16, 2008* F. M. auf der Heide and N. Shavit(Eds.,) ACM, pages 119–128, (2008).
4. A. Arora, S. Dolev, and M. Gouda. Maintaining digital clocks in step. *Parallel Processing Letters 1*, 1:11–18, (1991).
5. B. Awerbuch, and G. Varghese. Distributed program checking: a paradigm for building self-stabilizing distributed protocols (Extended Abstract). In: *Annual Symposium on Foundations of Computer Science (FOCS)*, San Juan, pages 258–267, 1991.
6. A. Beimel, and S. Dolev. Buses for anonymous message deliver. *Journal Cryptology*, 16(1):25–39, 2003.
7. A. Cournier, S. Devismes, and V. Villain. Light enabling snap-stabilization of fundamental protocols. *ACM Transactions on Autonomous and Adaptive Systems*, 4(1):27, 2009.
8. A. Czumaj, and C. Sohler. Testing expansion in bounded-degree graphs. In *FOCS* (2007), IEEE Computer Society, pages. 570–578.
9. E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications ACM 17*, 11 643–644, 1974.
10. S. Dolev. *Self-Stabilization*. MIT Press, Cambridge, MA, 2000.

11. S. Dolev, J.A. Garay, N. Gilboa, and V. Kolesnikov. In: *47th Annual Allerton Conference on Communication, Control, and Computing,* Monticello, IL, pages 1438–1445, 2009. Also brief Announcement in *ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 231–232, 2010.
12. S. Dolev, J. Garay, N. Gilboa, and V. Kolesnikov. Secret sharing Krohn-Rhodes: Private and perennial distributed computation. In: *Innovations in Computer Science (ICS)*, Beijing, China, January 2011.
13. S. Dolev, S. Gilbert, R. Guerraoui, F. Kuhn, and C. C. Newport. The wireless synchronization problem. In: *ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, Portland, pages 190–199, 2009.
14. S. Dolev, S. Gilbert, R. Guerraoui, and C. C. Newport. Gossiping in a multi-channel radio network. In: *International Symposium on Distributed Computing (DISC). Workshop on Distributed Algorithms (WDAG)*, pages 208–222, 2007.
15. S. Dolev, S. Gilbert, R. Guerraoui, and C. C. Newport. Secure communication over radio channels. In: *ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC),* Napoli, Italy, pages 105–114, 2008.
16. S. Dolev, S. Gilbert, N. A. Lynch, A. A. Shvartsman, and J. L. Welch. GeoQuorums: Implementing atomic memory in mobile *ad hoc* networks, *Distributed Computing 18* 2:125–155, 2005.
17. S. Dolev, S. Gilbert, L. Lynch, E. Schiller, A. A. Shvartsman, and J. L. Welch. Virtual mobile nodes for mobile Ad Hoc networks, In: *International Conference on Principles of DIStributed Computing*, (DISC 2007), pages 230–244.
18. S. Dolev, S. Gilbert, E. Schiller, A. A. Shvartsman, and J. L. Welch. Autonomous virtual mobile nodes, In *DIALM-POMC* (2005), pages 62–69.
19. S. Dolev, T. Herman, and L. Lahiani. Polygonal broadcast, secret maturity, and the firing sensors, *Ad Hoc Networks*, 4:447–486, 2006.
20. S. Dolev, L. Lahiani, and M. Yung. Secret swarm unitreactive k-secret sharing, In *INDOCRYPT*, pages 123–137, 2007.
21. S. Dolev, K. D. Pradhan, and J. L. Welch. Modified tree structure for location management in mobile environments, *Computer Communication* 19(4):335–345, 1996.
22. S. Dolev, M. Segal, H. Shpungin. Stretchable topology control, bounded-hop strong connectivity for flocking Swarms, 2009.
23. S. Dolev, E. Schiller, and J. L. Welch. Random walk for self-stabilizing group communication in ad hoc networks, *IEEE Transactions Mobile Computing 5*, (7):893–905, 2006.
24. S. Dolev, and N. Tzachar. Empire of colonies: Self-stabilizing and self-organizing distributed algorithms, *Theoretical Computer Science*, Volume 410 (6–7) pages 514–532, Special issue of OPODIS06, 2009.
25. S. Dolev and N. Tzachar. Spanders: Distributed spanning expanders. In: *Proceedings of the 25th ACM Symposium on Applied Computing (SAC-SCS)*, Nagoya, Japan, pages 1309–1314, 2010.
26. S. Dolev, and N. Tzachar. SPANDERS: Distributed spanning expanders. TR 08-02, Department of Computer Science, Ben-Gurion University of the Negev, 2007.
27. O. Goldreich, and D. Ron. On testing expansion in bounded-degree graphs. *Electronic Colloquium on Computational Complexity (ECCC) 7*:(20), 2000.
28. S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications, *Bulletin of the AMS*, volume 43, 4:439–561, 2006.
29. S. Kale, and C. Seshadrhi. Testing expansion in bounded degree graphs. ECCC report TR07–076, 2007.
30. L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM 21*, (7):558–565, 1978.
31. C. Law, and K.-Y. Siu. Distributed construction of random expander networks. In: *INFOCOM*, 2003.
32. L. Lovasz. Random walks on graphs: A survey.

33. R. Motwani, and P. Raghavan. *Randomized Algorithms*. Cambridge university press, Cambridge, 2006.
34. A. Nachmias, and A. Shapira. "Testing the expansion of graphs". ECCC report TR07–118, 2007.
35. M.K. Reiter, A. Samar, and C. Wang. Distributed construction of a fault-tolerant network from a tree. In: *SRDS 05: Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems,* IEEE Computer Society, Washington, DC, USA, pages 155–165, 2005.
36. S. A. Wright. "Registration of mobile packet data terminals after disaster,", *US Patent 6157633*, 1996.

Definitions Summary

Definition For a graph $G = (V, E)$, given two sets of nodes, V_1, V_2, define $E(V_1, V_2) = \{e = (v_1, v_2) \in E | v_1 \in V_1 \wedge v_2 \in V_2\}$ (e.g., the set of edges between V_1 and V_2). We also define $\overline{V_1} = V \setminus V_1$, the set of nodes not in V_1.

Definition For a graph $G = (V, E)$ and a given set of nodes, S, define $\Gamma(S) = \{u \in V | \exists s \in \wedge (u, s) \in E\}$.

Definition A graph $G = (V, E)$ is an *edge expander* if there exists a constant c, such that for each set S of vertices (where $|S| < |V|/2$) it follows that $|E(S, \overline{S})|/|S| > c$.

Definition An expander is considered "good" if it has a constant expansion parameter.

Definition A graph $G = (V, E)$ is a *vertex expander* if there exists a constant c, such that for each set S of vertices (where $|S| < |V|/2$) it follows that

$$P\left[\min_{S\subset V, |S|\le \frac{n}{2}} \frac{|\Gamma(S) \setminus S|}{|S|} < c\right] < o(1)$$

Definition Given a graph $G = (V, E)$, a *spander*, $\mathcal{S} = (V, E')$, is a spanning subgraph of G if there exists a constant $p > 0$, such that $|E'| \le p|E|$ and the edge expansion of $\mathcal{S}$ is at worst p times the edge expansion of G.

Definition The *mixing rate* of a graph is the measure of how fast a random walk on the graph converges to its stationary distribution.

Definition The *mixing time* of a graph gives the time scale (in steps) for a random walk to reach the stationary distribution.

Definition A *task* is defined by a set of *legal executions*.

Definition A *fair execution* is an execution of the system in which every node makes steps infinitely often.

Definition A configuration c is a *safe configuration* for a system and a set of legal executions LE if every fair execution that starts in c is in LE.

Definition A system is self-stabilizing for a task and a set of legal executions LE if every infinite execution reaches a safe configuration in relation to LE.

Definition A *communication round* (or just a round) is a sequence of atomic steps such that each node has taken at least one atomic step during this sequence. If this atomic step involves a send operation of a message m over link l, then we require that the atomic step which corresponds to receiving a message from l, which was sent during this sequence of atomic steps, will also appear in the sequence.

Definition A distributed algorithm is termed *self-organizing* ([24]) if it satisfies the following properties: (1) the algorithm is self-stabilizing, (2) convergence time to a safe configuration, $s(n)$, is in $o(n)$, and (3) after reaching a safe configuration, convergence time following a dynamic change, $d(n)$, is in $o(s(n))$.

Definition A distributed algorithm is termed *snap-stabilizing* if the algorithm stabilizes following the first request by any node and before, or simultaneously with, a notification arriving to the requesting node at the completion of the request (for more information, see [7]).

Chapter 21
Computing by Mobile Robotic Sensors

Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro

Abstract The research areas of *mobile robotic sensors* lie in the intersection of two major fields of investigations carried out by quite distinct communities of researchers: *autonomous robots* and *mobile sensor networks*. Robotic sensors are micro-robots capable of locomotion and sensing. Like the sensors in wireless sensor networks, they are *myopic*: their sensing range is *limited*. Unlike the sensors in wireless sensor networks, robotic sensors are *silent*: they have no direct communication capabilities. This means that synchronization, interaction, and communication of information among the robotic sensors can be achieved solely by means of their sensing capability, usually called *vision*. In this chapter, we review the results of the investigations on the computability and complexity aspects of systems formed by these *myopic* and *silent* mobile sensors.

21.1 Introduction

21.1.1 Distributed Computing and Mobile Entities

In distributed computing the research focus is on the computational and complexity issues of systems composed of autonomous computational entities interacting with each other (e.g., to solve a problem, to perform a task). While traditionally the entities have been assumed to be *static*, recent advances in a variety of fields, ranging from robotics to artificial intelligence to software engineering to networking, have motivated the distributed computing community to address the situation of *mobile* entities. Indeed, recently an increasing number of investigations are being carried out on the computational and complexity issues arising in systems of autonomous mobile entities located in a spatial universe $\mathcal{U}$. The entities have storage and processing capabilities, exhibit the same behavior (i.e., execute the same protocol), and can move in $\mathcal{U}$ (their movement is constrained by the nature of $\mathcal{U}$).

P. Flocchini (✉)
School of Information Technology and Engineering, University of Ottawa, Ottawa, ON, Canada
e-mail: flocchin@site.uottawa.ca

Depending on the nature of $\mathcal{U}$, there are two basic settings in which autonomous mobile entities are being investigated. The first setting, sometimes called *graph world* or *discrete universe*, is when the universe is a simple graph; this is, for example, the case of mobile agents in communication networks (e.g., [15, 26, 36]). The second setting, called sometimes *continuous universe*, is when $\mathcal{U}$ is a region of the 2D (or 3D) space. This is, for example, the case of robotic swarms, mobile sensor networks, mobile robotic sensors, etc. (e.g., [1, 13, 16, 21, 31, 34, 35, 38, 45, 47, 48, 56, 66, 75, 79, 81–83]). In both settings, the research concern is on determining what tasks can be performed by such entities, under what conditions, and at what cost. In particular, a central question is to determine what minimal hypotheses allow a given problem to be solved.

In the continuous setting two major research areas can be distinguished, their difference resting on the types of assumptions made, carried out by quite distinct communities of investigators:

- *autonomous robots*, an established and mature research field (which includes swarm robotics and robotic networks), investigated mainly by researchers in robotics, control, artificial intelligence, and more recently by algorithmic researchers;
- *mobile sensor networks*, a new and emerging research field whose investigations are carried out mostly as an extension of the more traditional (static) sensor networks.

These two areas have overlapping boundaries, and their intersection is a region of surprising and interesting convergence of research interests. An important such region is the area of *mobile robotic sensors*, the topic of this chapter.

21.1.2 Robots, Sensors, and Mobility

The addition of motorial capabilities to a computational entity not only empowers the entity in non-trivial ways, but it also, and more importantly, empowers the system employing such entities. This empowerment takes many forms and displays different aspects. This is particularly evident in the case of *wireless sensor networks*. Indeed, empowering the sensors with mobility allows the network to perform tasks and solve problems which would be impossible to do with static sensors. Indeed, mobile sensors have gained attention lately as important tools for a wide range of applications and tasks, such as search and rescue, exploration and mapping, evaluation of civil infrastructure, military operations.

The first proposals for the use of mobility in sensor networks have been for *exogenous* solutions: mobile robots are introduced into the network of static sensors to augment the capacities of the system or to simplify the management of the network (e.g., to repair failed sensors, to redeploy sensors so to improve overall coverage, to gather information from the robots), and this research still continues (e.g., see [5, 6, 27, 54, 67, 84, 86]).

At the same time, a large number of investigators have suggested and studied the use of *endogenous* mobility in sensor networks: *mobile sensor networks*, in which the sensors have processing power, wireless communication, and motion capabilities. The sensors operate in a totally distributed way, moving under their own control, and reacting to the inputs received from the environment where they operate, thus creating computational systems capable of interacting with the physical environment (e.g., see [11, 14, 44–46, 50, 57, 58, 61, 71, 82, 88]).

At this point, the research results start to merge with the rich existing literature on *autonomous robots*, in particular with that of systems of *micro-robots* and of *robotic sensors*, studied also from the control and the computing point of view (e.g., [1, 7, 13, 18, 21, 28, 31, 34, 38, 56, 66, 76, 82, 83]). These investigations differ greatly from each other depending on the assumptions they make. Major differences exist depending on whether the entities' actions are synchronized (e.g., [13, 47, 81, 83]) or no timing assumptions exist (e.g., [31, 35, 53]); the sensors have persistent memory (e.g., [13, 43, 83]) or are oblivious (e.g., [35, 46, 79]); the sensors have the computational power of Turing machines (e.g., [81, 83]) or are simple Finite-State machines (e.g., [4, 16, 30, 47]); the visibility/communication range is limited (e.g., [31, 38, 43, 47, 79]) or extends to the entire region (e.g., [1, 10, 12, 35, 83]).

21.1.3 Mobile Robotic Sensors

The crucial difference between systems of *robotic sensors* and wireless *sensor networks* is the following. In *sensor networks*, regardless of whether mobile or static, the entities are endowed with both *sensing* and (wireless) *communication* capabilities. However, of these two capabilities, only the latter—wireless communication — is used for synchronization, interaction, and communication of information among the sensors and within the network.

On the contrary, *robotic sensors* are generally endowed solely with *sensing* capability. In other words, they are *silent*: they have no direct communication capabilities. This means that synchronization, interaction, and communication of information among the sensors and within the network can be achieved solely by means of their sensing capability, usually called *vision*. This characteristic is indeed the same one assumed in most of the traditional research on autonomous robots.[1]

The lack of direct means of communication has many computational drawbacks; indeed, the fact that robotic sensors must rely solely on their sensing capabilities for all their interactions is a severe limitation. It does however have one advantage in the determination of an entity's neighbors. In fact, in systems of robotic sensors, the determination of one's neighbours is done by sensing capabilities (e.g., vision): any sensor in the sensing radius is detected even if inactive, and thus no other mechanisms are needed. On the other hand, in traditional wireless sensor

[1] The notable exceptions are the *robotic networks* studied in the control community that assume and use direct communication, e.g., [7, 65].

networks, determination of the neighbors is achieved by radio communication; since an inactive sensor does not participate in any communication, the simple activity of determining one's neighbors, to be completed, requires the use of randomization or the presence of sophisticated synchronization and scheduling mechanisms (e.g., [63, 69, 74]).

Another important difference with mobile sensor networks is that robotic sensors, like most autonomous robots, are often equipped with a much larger energy reserve, or have self-charging capability (e.g., on board PV or ability to plug into the power grid to recharge their batteries). Hence energy is a concern but not as crucial as in mobile sensor networks.

The key feature that robotic sensors share with mobile sensor networks is that they are *myopic*: their sensing range is *limited* (see Fig. 21.1). Precisely this feature constitutes the key difference between robotic sensors and traditional models of autonomous robots and micro-robots. In fact, algorithmic robotic research usually assumes *unlimited visibility*: the entities are capable of determining the location of all other regardless of their position in the region, e.g., [1, 9, 12, 19, 35, 52, 53, 70, 77, 81, 83, 91]. Additional differences between robotic sensors and traditional models of autonomous robots and micro-robots robotic sensors are that usually the robots are more powerful (both memory-wise and computationally) than sensors, and typically there is no requirement for the robots to reach a state of static equilibrium (e.g., in most cases the swarm just converges toward a desired formation or pattern).

Summarizing, robotic sensors are

1. *mobile*, like mobile sensor networks and autonomous robots;
2. *silent*, like traditional autonomous robots;
3. *myopic*, like sensor networks.

The purpose of this chapter is to present and discuss the research efforts on systems of robotic sensors. It is organized as follows. In Sect. 21.2 we will present in detail the computational model and introduce the formalism and terminology. We will then review the research results on computing by mobile robotic sensors. The investigations have been focusing on three fundamental problems: *Self-Deployment*,

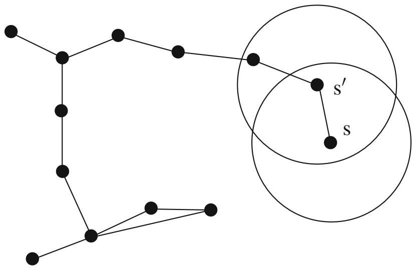


Fig. 21.1 Limited visibility: a sensor can only see sensors that are within its radius of visibility

Pattern Formation, and *Gathering*; they will be discussed in Sects. 21.3, 21.4, and 21.5, respectively.

21.2 Modeling Mobile Robotic Sensors

21.2.1 Capabilities

The system is composed of a set $\mathbb{S} = \{s_1, \ldots, s_n\}$ of n mobile robotic sensors operating in a spatial universe $\mathbb{U} \subseteq \mathbb{R}^2$.

A *mobile robotic sensor* (or simply *sensor*) $s \in \mathbb{S}$ is modeled as a computational unit: it has its own local memory and it is capable of performing local computations.

A sensor is endowed with sensorial capabilities and it can perceive the spatial environment $\mathbb{U}$ and the sensors in it, within a fixed distance $v > 0$, called *visibility radius*. Each sensor has its own local coordinate system: a unit of length, an origin, and a Cartesian coordinate system defined by the *directions* of two coordinate axes, identified as the x- and y-axis, together with their *orientations*, identified as the positive and negative sides of the axes. However, the local coordinate systems of the sensors might not be consistent with each other.

Each sensor is endowed with motorial capabilities; it can turn and move in any direction. A move may stop before the sensor reaches its destination, e.g., because of limits to its motion energy; however, it is assumed that the distance traveled in a move by s is not infinitesimally small (unless it brings the sensor to its destination): there exists a constant $\delta_s > 0$, such that, if the destination is closer than δ_s, s will reach it; otherwise, s will move toward it by at least δ_s. Note that without this assumption, it would be impossible for s to ever reach its destination, following a classical Zenonian argument. In the following, we shall use $\delta = \min_s \delta_s$.

The sensors are *silent*: they have no means of direct communication of information to other sensors. Thus, any communication occurs in a totally implicit manner, by observing the other sensors' positions. Let $s(t)$ denote the position of sensor s at time t; when no ambiguity arises, we shall omit the temporal indication.

The sensors are *autonomous* (i.e., without a central control) and *identical* (i.e., they execute the same protocol). They might be *anonymous* (i.e., a priori indistinguishable by their appearance and without identifiers that can be used during the computation).

21.2.2 Behavior

At any point of time, a sensor is either *active* or *inactive*. When *active*, a sensor s performs the following three operations, each in a different state:

1. (State *Locate*) It observes the spatial environment $\mathbb{U}$ and the sensors in it, within its visibility radius $v > 0$. As a result, it determines, in its own coordinate system,

a snapshot of the positions of the sensors in its *circle of visibility* at that time. The *circle of visibility* of s at time t is the surrounding circle of s in its most recent *Locate*. The *Locate* state can be assumed, without loss of generality, to be instantaneous.[2]

2. (State *Compute*) It performs a local computation, according to an algorithm (the same for all sensors) that takes in input the result of its previous *Locate*, and returns a destination point. Hence, the algorithm, the same for all sensors, will specify which operations a sensor must perform whenever it is active.
3. (State *Move*) It moves toward the computed destination point; if the destination point is the current location, the sensor stays still. A move may stop before the sensor reaches its destination, e.g., because of limits to the sensor's motion energy.

When *inactive* a sensor is in *Sleep* state:

4. (State *Sleep*) It is idle and does not perform any operation.

In summary, the sensors operate in a continuous *Locate-Compute-Move-Sleep* life cycle.

21.2.3 Synchronization

Depending on the degree of synchronization among the life cycles of different sensors, three sub-models are traditionally identified: *synchronous*, *semi-synchronous*, and *asynchronous*.

In the *synchronous* model (SYNC), the cycles of all sensors are fully synchronized: the sensors become active all at the same time and each operation of the life cycle is performed by all sensors simultaneously. Alternatively, there is a global clock tick reaching all sensors simultaneously, and a sensor's cycle is an instantaneous event that starts at a clock tick and ends by the next. As a consequence, no sensor will ever be observed while moving. This model is used, e.g., in [21, 38, 83].

In the *semi-synchronous* model (SSYNC), there is a global clock tick reaching all sensors simultaneously, and a sensor's activities are an instantaneous event that starts at a clock tick and ends by the next. Hence, also in this model, no sensor will ever be observed while moving. However, at each clock tick, some sensors might not become active. The unpredictability of which sensors become active at a clock tick is restricted by the fact that at every clock tick at least one sensor is active, and every sensor becomes active infinitely often. This model, sometimes called ATOM, is used, e.g., in [1, 9, 12, 13, 19, 21, 83].

In the *asynchronous* model (ASYNC), there is no global clock and the sensors do not have a common notion of time. Furthermore, the duration of each activity (or

[2] Any time spent to activate its sensors (before the snapshot is taken) and to process the information retrieved with the snapshot will be charged to the *Sleep* and the *Compute* state, respectively.

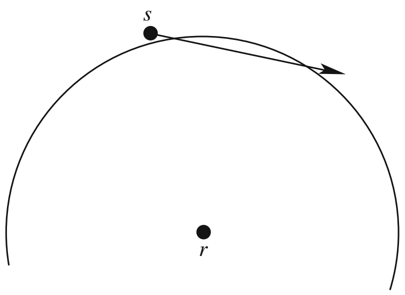


Fig. 21.2 When s starts moving (the *left* end of the arrow), r and s do not see each other. While s is moving, r enters state *Locate* and sees s; however, s is still unaware of r. After s passes the visibility *circle* of r, it is still unaware of r

inactivity) is finite but unpredictable. As a result, sensors can be seen while moving, and computations can be made based on obsolete observations. For example (see Fig. 21.2), sensor s in transit toward its destination is seen by r; however, s is not aware of r's existence and, if it starts the next cycle before r starts moving, s will continue to be unaware of r. This (realistic but more difficult) model, sometimes called CORDA, is used, e.g., in [9, 33–35, 52, 53, 70].

21.2.4 Memory

In addition to its programs, each sensor has a local working memory, or *workspace*, used for computations and to store different amount of information (e.g., regarding the location of its neighbors) obtained during the cycles. Two sub-models have been identified, depending on whether or not this workspace is persistent.

In the *oblivious* model, all the information contained in the workspace is *cleared* at the end of each cycle. In other words, the sensors have *no* memory of past actions and computations, and the computation is based solely on what is determined in the current cycle. The importance of obliviousness comes from its link to *self-stabilization* and *fault tolerance*. This model, sometimes improperly called *memoryless*, is used, e.g., in [9, 12, 13, 19, 34, 35, 53].

In the *persistent memory* model, all the information contained in the workspace is *legacy*: unless explicitly erased by the sensor, it will persist throughout the sensor's cycles. This model is commonly used for both wireless sensor networks and micro-robots.

An additional important parameter is the *size* of the persistent workspace. Noticeable are the two extreme cases. One extreme is the *unbounded memory* case, where no information is ever erased; hence sensors can remember all past computations and actions (e.g., see [81, 83]). The other extreme is when the size of the workspace is constant; in this case, the sensors are just *Finite-State Machines* (e.g., [4, 16, 47]) .

21.3 Self-Deployment

21.3.1 Introduction

The first important problem faced with sensor systems is the effective deployment of the sensors within the spatial universe $\mathbb{U}$, assumed to be *finite*. The deployment must usually satisfy some optimization criteria with respect to the space $\mathbb{U}$ (e.g., uniformity, maximum coverage). In case of static sensors, they are usually deployed by external means, either carefully (e.g., manually installed) or randomly (e.g., dropped by an airplane); in the latter case, the distribution of the sensors may not satisfy the desired optimization criteria.

If the sensing entities are *mobile*, as in the case of mobile sensor networks, vehicular networks, and robotic sensor networks, they are potentially capable to position themselves in appropriate locations without the help of any central coordination or external control, a task called *Self-Deployment*.

In this section we consider some of the problems and issues we must face to achieve such a rather complex task; indeed, designing localized algorithms for efficient and effective deployment of the mobile entities is a challenging research issue.

Some of the initial proposals on the deployment of mobile sensors were still based on centralized approaches, e.g., employing a powerful cluster head to collect the initial location of the mobile sensors and determine their target location [92]. The current research efforts are on the development of local protocols that allow the sensors to move from an initial random configuration to a uniform one acting in a purely local, decentralized, distributed fashion. An essential requirement is clearly that the sensors will reach a state of *static equilibrium*, that is, the self-deployment will be completed within finite time. How this task can be efficiently accomplished continues to be the subject of extensive research in the mobile sensor networks community (e.g., see [43–46, 58, 62, 72, 87, 88]). Similar questions have been posed in terms of *scattering* or *coverage* in cooperative mobile robotics and swarm robotics (e.g., [8, 47]), as well as in terms of the *formation* problem for those entities (e.g., [9, 13, 19, 33, 35, 53, 81, 83, 85]).

The existing self-deployment protocols differ greatly from each other depending on the assumptions they make; for example, some require the sensors to be deployed one at a time [16, 45, 47], while others require prespecified destinations for the sensors [62]. However, sensors are usually dispersed in the environment all together, more or less at the same time, with no a priori knowledge of where their final location should be. Actually, unlike the case of ad hoc networks, for small sensors localization is very hard, so it cannot be generally assumed that the sensors know where they are.

The self-deployment problem has been investigated with the goal to cover the area so to satisfy some optimization criteria, typically to maximize the coverage (e.g., see [43–46, 58, 62, 72, 87, 88]). For example, in [88] the problem is to maximize the sensor coverage of the target area minimizing the time needed to cover the area, while in [72] the additional constraint is a minimum requirement on the degree of all nodes. Typically, distributed self-deployment protocols first discover the existence of *coverage holes* (the area not covered by any sensor) in the target area based on the sensing service required by the application. After discovering a coverage hole, the protocols calculate the target positions of these sensors, that is, the positions where they should move.

All of these solutions to the self-deployment problem require direct communication between sensors, hence cannot be employed by robotic sensors. It is also interesting to observe that even with communication, none of the existing self-deployment proposals is capable of providing a complete uniform coverage. This impossibility is hardly surprising since those protocols are *generic*, that is, they must work in any environment regardless of its topology or structure. This fact opens a series of interesting questions, first of all whether it is possible for the sensors to self-deploy achieving uniform coverage in specific environments (e.g., corridors, grids, rims). The next important question is on the capabilities and a priori knowledge needed by the sensors to achieve this goal; in other words, how "weak" the sensors can be and still be able to uniformly self-deploy. In particular, the focus of this section is on conditions for self-deployment of mobile robotic sensors so as to obtain uniform coverage of specific spaces $\mathbb{U}$.

21.3.2 Uniform Deployment on Linear Borders

The first spatial universe $\mathbb{U}$ considered is possibly the simplest: a *linear border* or *corridor*, along which the sensors are required to place themselves evenly.

A corridor can be viewed as a *line* $\mathcal{L}$ on which the sensors are initially located at random distinct points. From an external point of view, the sensors can be ordered based on their position on the line from left to right; without loss of generality, let s_1 be the leftmost sensor and let s_i be to the right of s_{i-1}, $2 \leq i \leq n$. This order is clearly unknown to the sensors.

The goal is for the sensors to self-deploy evenly in the segment of the line delimited by the positions of the leftmost and rightmost sensors s_1 and s_n (that, alternatively, could represent some perimeter marks rather than sensors).

Assuming that each sensor s_i is capable of viewing its neighbors s_{i-1} and s_{i+1} if they exist, the self-deployment algorithm, by Cohen and Peleg [13], is remarkably simple:

Protocol CORRIDOR SPREAD (for sensor s_i)

- If no other sensor is seen on the left or on the right, then do nothing;
- Otherwise, move to point $x = \frac{1}{2}\,(s_{i+1} + s_{i-1})$.

With sensors that are anonymous, oblivious, and with no common coordinate system, the above protocol *converges* to a uniform deployment in the SSYNC (and thus also in the SYNC) model.

Let us show the idea of the convergence proof in the SYNC model. Note that since the sensors operate in one dimension, any coordinate system will give the same resulting destination. Therefore, in order to analyze the protocol, an external global coordinate system is used, of which the sensors have clearly no knowledge. In the following, the coordinate system where $s_0(t) = 0$ and $s_{n-1}(t) = 1$ is chosen as the global coordinate system. The goal is to spread the sensors uniformly; that is, at the end, sensor s_i should occupy position $\frac{i}{n-1}$. Let $\mu_i[t]$ be the *shift* of the s_i's location at time t from its final position. According to the protocol, the position of sensor s_i changes from $s_i(t)$ to

$$s_i(t+1) = \frac{1}{2}\left(s_{i-1}(t) + s_{i+1}(t)\right)$$

for $2 \le i \le n-1$, while sensors s_1 and s_n never move. Therefore, the shifts change with time as

$$\mu_i[t+1] = \frac{1}{2}\left(\mu_{i+1}[t] + \mu_{i-1}[t]\right)$$

Consider the following *progress* measure:

$$\psi[t] = \Sigma_{i=1}^{i=n}\, \mu^2[t]$$

Then

Lemma 1 *$\psi[t]$ is a decreasing function of t unless the robots are already equally spread.*

Finally,

Theorem 1 *In the* SYNC *model, every $O(n^2)$ cycles, $\psi[t]$ is at least halved; furthermore, the sensors converge to equidistant positions.*

The idea of the convergence prove in SSYNC is similar; in fact, first a non-decreasing quantity is defined, and its monotonicity proven. Then, by relating this quantity to the non-constant terms of the cosine series, it is proven that it decreases by a constant factor on every round, proving convergence.

Theorem 2 *In the* SSYNC *model, anonymous, oblivious sensors on a line $\mathcal{L}$ with no common coordinate system* converge *to uniform deployment.*

21.3.3 Uniform Deployment Along Circular Borders

Consider next an important class of spatial regions, that of *circular borders* or *circular rims*. Deployment in these spaces occurs, for example, when the sensors have to surround a dangerous area and can only move along its outer perimeter. This situation is modeled by describing the space $\mathbb{U}$ as a *ring* $\mathcal{C}$. Starting from an

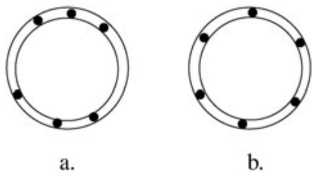


Fig. 21.3 Starting from an initial arbitrary placement (**a**), the sensors must move to a uniform cover of the ring (**b**)

initial arbitrary placement on the ring, the sensors must within finite time position themselves along the ring at (approximately) equal distance, see Fig. 21.3.

The self-deployment problem along a ring is related to the well-studied problem in the field of swarm robotics of *uniform circle formation* [9, 19, 22, 23, 53, 77, 85]. In this problem (discussed in more detail in Sect. 21.4.2), the robots are required to uniformly place themselves on the circumference of a circle not determined in advance (i.e., the entities do not know the location of the circle to form). The main difference between the uniform circle formation and the self-deployment problem in the ring is that in uniform circle formation the entities can freely move on the two-dimensional plane in which they have to form a ring; in contrast, our sensors can move only *on* the ring, which is the entire environment.

Let $\mathbb{S} = \{s_1, \ldots, s_n\}$ be the n sensors initially arbitrarily placed on the ring $\mathcal{C}$ (see Fig. 21.3). Initially no two sensors are placed at the same location; the algorithms should avoid *collisions*, i.e., having two sensors simultaneously occupying the same point; without loss of generality, let s_i be the sensor immediately before s_{i+1} in the clockwise direction, with s_n preceding s_1. Let $d_i(t)$ be the distance between sensor s_i and sensor s_{i+1} at time t; when no ambiguity arises, we will omit the time and simply indicate the distance as d_i. Let $d = L/n$, where L denotes the length of the ring $\mathcal{C}$. The sensors have reached an *exact* self-deployment at time t if $d_i(t) = d$ for all $1 \leq i \leq n$. Given $\varepsilon > 0$, the sensors have reached an ε*-approximate* self-deployment at time t if $d - \varepsilon \leq d_i(t) \leq d + \varepsilon$ for all $1 \leq i \leq n$.

An algorithm $\mathcal{A}$correctly solves the *exact* (resp. ε*-approximate*) self-deployment problem if, in any execution of $\mathcal{A}$by the sensors in $\mathcal{C}$, regardless of their initial position in $\mathcal{C}$, there exists a time t' such that the sensors have reached an *exact* (resp. ε*-approximate*) self-deployment at time t' and are in a quiescent state.

21.3.3.1 Impossibility Without Orientation

There is a strong negative result for the SSYNC (and thus for the ASYNC) model. In fact, *exact* self-deployment is actually *impossible* if the sensors do not share a *common orientation* of the ring; notice that this is much less a requirement than having global coordinates or sharing a common coordinate system. This impossibility result by Focchini et al. [31] holds even if the sensors have unlimited memory of the past computations and actions (i.e., unlimited persistent memory, see Sect. 21.2.4), and their visibility radius is unlimited.

Theorem 3 *Let the sensors be on a ring $\mathcal{C}$. In the absence of common orientation of $\mathcal{C}$, there is no deterministic exact self-deployment algorithm even if the sensors have* *unbounded persistent memory, their visibility radius in unlimited. and the scheduling is* SSYNC.

To see why this is the case, consider the following setting. Let n be even; partition the sensors into two sets, $S_1 = \{s_1, \dots, s_{n/2}\}$ and $S_2 = S \setminus S_1$, and place the sensors of $S_1$and S_2on the vertices of two regular $(n/2)$-gons on *Circle*, rotated of an angle $\alpha < 360°/n$. Furthermore, all sensors have their local coordinate axes rotated so that they all have the same view of the world (refer to Fig. 21.4.a for an example). In other words, the sensors in S_1share the same orientation, while those in S_2share the opposite orientation of $\mathcal{C}$. Denote a configuration with such properties by $Y(\alpha)$. A key property of $Y(\alpha)$ is the following.

Property 1 Let the system be in a configuration $Y(\alpha)$ at time step t_i.

1. If activating only the sensors in S_1, *no* exact self-deployment on $\mathcal{C}$ is reached at time step t_{i+1}, then also activating only the ones in S_2 *no* exact self-deployment on $\mathcal{C}$ would be reached at time step t_{i+1}; furthermore, in either case the system would be in a configuration $Y(\alpha')$ for some $\alpha' < 360°/n$.
2. If activating only the sensors in S_1 an exact self-deployment on $\mathcal{C}$ is reached at time step t_{i+1}, then also activating only the sensors in S_2an exact self-deployment on $\mathcal{C}$ would be reached at time step t_{i+1}.
3. If activating only the sensors in S_1 an exact self-deployment on $\mathcal{C}$ is reached at time step t_{i+1}, then activating both sets no exact self-deployment on $\mathcal{C}$ would be reached at time step t_{i+1}, and the system would be in a configuration $Y(\alpha')$ for some $\alpha' < 360°/n$.

Using this property it is easy to design an adversary that will force any self-deployment $\mathcal{A}$ to never succeed in solving the problem: the adversary will choose $Y(\alpha)$ as the initial configuration and behave as follows (refer to Figure 21.5):
(Step a) If activating only the sensors in S_1 no exact self-deployment on $\mathcal{C}$ is reached: then activate all sensors in S_1, while all sensors in S_2 are inactive; otherwise, activate all sensors. Go to (b).

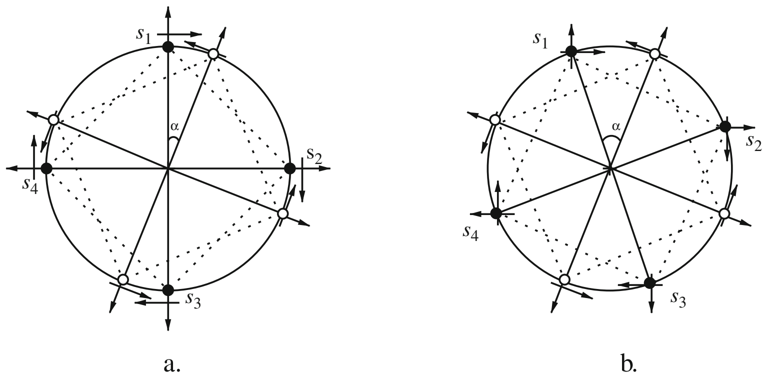


Fig. 21.4 (**a**) An example of starting configuration for the proof of Theorem 3. The *black* sensors are in S_1, while the *white* ones in S_2. (**b**) Theorem 3: the adversary moves only sensors in S_1

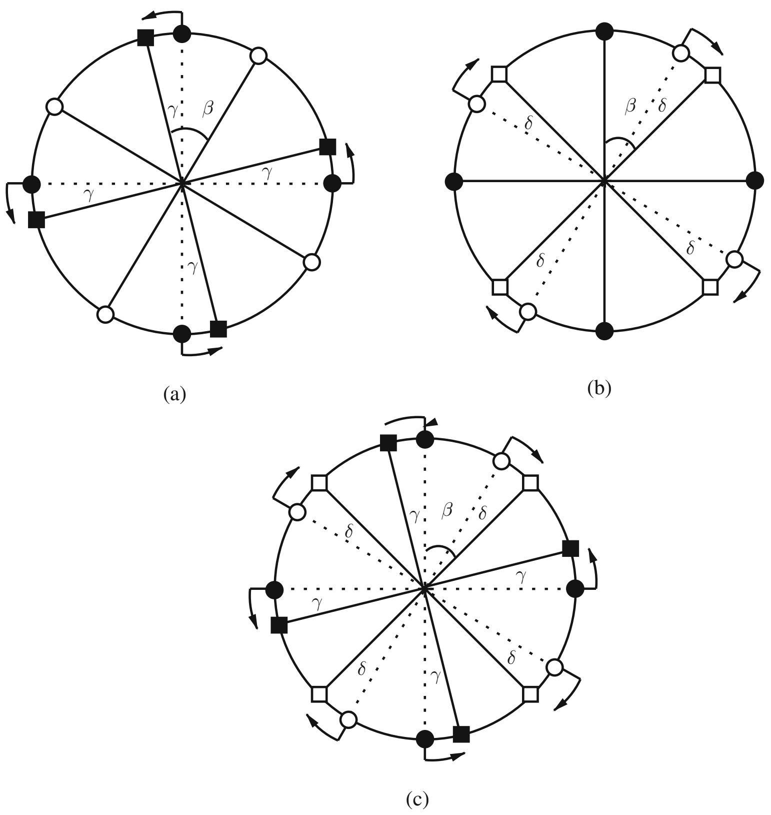


Fig. 21.5 Theorem 3. (**a**) If only the sensors in $S_1$are activated at t, all sensors would be uniformly placed at time $t + 1$, with $\beta + \gamma$ =45°. (**b**) If only the sensors in $S_2$are activated at t, all sensors would be uniformly placed at time $t + 1$, with $\beta + \gamma = 45°$. (**c**) Therefore, if all sensors would be activated at t, they would not be in an exact self-deployment on $\mathcal{C}$, having $\gamma+\beta+\delta \neq 2\pi/n = 45°$. In all figures, the *squares* represent the destination of the active sensors

(Step b) If activating only the sensors in S_2 no exact self-deployment on $\mathcal{C}$ is reached: then activate all sensors in S_2, while all sensors in $S_1$are inactive; otherwise, activate all sensors. Go to (a).

By Property 1, if the configuration at time $t_i \geq t_0$ is $Y(\alpha)$ for some $\alpha < 360°/n$, then regardless of whether the adversary executes step (a) or (b), the resulting configuration is $Y(\alpha')$ for some $\alpha' < 360°/n$, and hence *no* exact self-deployment on $\mathcal{C}$ is reached at time step t_{i+1}. Hence, there exists an infinite execution of $\mathcal{A}$ in which no exact self-deployment will ever be reached. The alternating between steps (a) and (b) by the adversary ensures the feasibility of this execution: every sensor will in fact become active infinitely often.

Recently, the impossibility without orientation has been announced to hold also for the stronger SYNC model [29].

Since the impossibility result of Theorem 3 holds in the absence of common orientation of the ring, the focus will now be on *oriented* rings; two cases will be considered depending on whether or not the desired final distance d is known to the sensors.

21.3.3.2 Exact Deployment

Faced with this strong negative result of Theorem 3, the interesting question becomes under what restrictions the self-deployment problem can be solved with an exact algorithm. Since the impossibility result holds in the absence of common orientation of the ring, consider the problem in *oriented* rings.

In an oriented ring, if the desired final distance d is known or computable (e.g., both the number of sensors and the length of the ring are known), *exact* self-deployment is indeed possible. This positive result holds even if the sensors are oblivious and asynchronous, provided their visibility radius is at least $2d$.

The algorithm by Focchini, Prencipe, and Santoro [31] proves this result is very simple:

Protocol RING - KNOWN INTERDISTANCE (for sensor s_i)

- Locate clockwise at distance $2d$. Let d_i be the distance to s_{i+1} (if visible, else $d_i = 2d$).
- If $d_i \leq d$ do not move.
- If $d_i > d$ move clockwise and place yourself at distance d from s_{i+1} (if visible, else at distance d from current location).

Like in other cases (e.g., [12, 13]), the difficulty is not in the protocol but in the proof of its correctness. Using this protocol, and observing that the algorithm operates in ASYNC, we have

Theorem 4 *Let the sensors share a common orientation of the ring $\mathcal{C}$ and be able to locate to distance $2d$. If they know d, then* exact *self-deployment is possible even if the sensors are oblivious and the scheduling is* ASYNC.

21.3.3.3 ε-Approximate Deployment

In an oriented ring, if the sensors do *not* know the desired final distance d, then ε-approximate self-deployment is still possible for any $\varepsilon > 0$; also in this case, the protocol works even for the weakest sensors: oblivious and asynchronous, provided their visibility radius is greater than $2d$.

Also in this case the proof is provided by a simple protocol [31]: sensors asynchronously and independently locate in both directions at distance v, then they position themselves in the middle between the closest observed sensor (if any).

Protocol RING - UNKNOWN INTERDISTANCE (for sensor s_i)

- Locate around at distance v. Let d_i be the distance to next sensor, d_{i-1} the distance to the previous (if no sensor is visible clockwise, $d_i = v$, analogously for counterclockwise).
- If $d_i \le d_{i-1}$ do not move.
- If $d_i > d_{i-1}$ move to $\frac{d_i + d_{i-1}}{2} - d_{i-1}$ clockwise.

This algorithm converges to a uniform deployment. The crucial property is that

Property 2 For any $\varepsilon > 0$ there exists a time t, such that $\forall t' > t, \forall i\colon |d_i(t') - d| \le \varepsilon$.

Hence, by adding to the protocol a test on whether both d_i and d_{i-1} are within ε from d (in which case no move is performed by s_i), it follows that *ε-approximate* self-deployment is possible even if the scheduling is ASYNC, if the sensors share a common orientation of the ring $\mathcal{C}$ and are able to locate to distance $v > 2d$.

The strategy used by the protocol described here is *go-to-half*. Interestingly, it was shown by Dijkstra ([25] pp. 34–35) that in an unoriented ring *go-to-half* does *not* converge, and hence cannot be used for approximate self-deployment in an unoriented ring. However, a different strategy, *go-to-quarter*, does converge in an unoriented ring [19, 77] and can thus be used for *ε-approximate* self-deployment in an unoriented ring with unknown d, yielding the following more powerful result:

Theorem 5 *Let the sensors in the ring $\mathcal{C}$ be able to locate to distance $v > 2d$. Then* ε-approximate *self-deployment is possible even if the sensors are oblivious, the scheduling is* ASYNC, *and the ring is not oriented.*

21.3.4 Uniform Deployment in Rectangular Spaces

21.3.4.1 Problem Definition and Notation

The next class of spaces $\mathbb{U}$ considered is that of *rectangular spaces*, that is, spaces delimited by a rectangular *border* B. The sensors are capable of detecting any part of B within their visibility radius. Assume that there is a local sense of orientation: each sensor has a consistent notion of "up-down" and "left-right" (e.g., as provided by a compass), where the "up-down" axis is parallel to the longest side of the border.

A rectangular space of size $L \times W$ can be logically subdivided into equal sized square of size d^2 by considering a $(l + 1) \times (w + 1)$ rectangular *grid* $\mathcal{G}$, where $l = L/d$ and $w = W/d$ and the distance between neighboring nodes is d, and the visibility radius is $v \ge 2d$. The uniform self-deployment problem in rectangular spaces thus consists of reaching an equilibrium configuration where the sensors are evenly placed among the grid points (see Fig. 21.6).

Let us indicate by $(0, 0)$ the leftmost lower corner of $\mathcal{G}$ and by (i, j) the node belonging to column i and row j. For simplicity, assume $n = (k + 1)^2$, that $\mathcal{G}$ is a square grid with $l = w = k \cdot d$, and that the sensors are initially arbitrarily located at distinct grid points. In this case, an equilibrium configuration consists of nodes

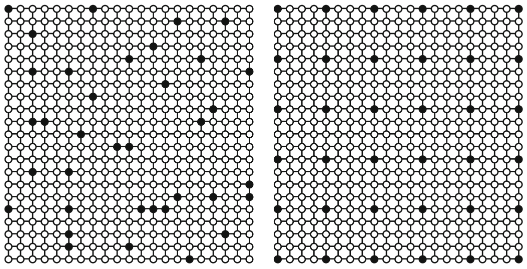


Fig. 21.6 A random initial configuration and a uniform deployment

$(i \cdot d, j \cdot d)$, with $i, j \in [0, k]$ hosting exactly one sensor each (see Fig. 21.6). The goal is to design a collision-free protocol for positioning the sensors at those grid points. Note that the (i, j) coordinates of the grid nodes are global and thus *not known* to the sensors, which use only relative coordinates. The common orientation will be modeled by assuming that the edges of the grid are consistently labeled *Up*, *Down*, *Left*, and *Right*, and edge labels are visible to the sensors.

21.3.4.2 Uniform Deployment Protocol

The uniform deployment algorithm SCATTER is a set of local rules for the robots designed by Barriére et al. [4]. Each sensor has a *state* variable belonging to a set of states $\{-1, 0, 1, 2, 3, 4\}$ (initialized to -1) which determines the set of rules to be followed, which solely depend on the robot's current state, its position, and the positions of the robots within its visibility radius.

Upon start-up, the execution of the algorithm is logically divided into three *phases*: *Cleaning*, *Collecting*, and *Deploying*. Waking up for the first time (in state -1), each sensor determines what phase to start, depending on its relative position. If a sensor is at the *left-upper corner* it directly enters state 3 (i.e., it starts *Deploying*). If a robot is on the *left or bottom border*, it enters state 0 (i.e., it starts *Cleaning*). Otherwise, it enters state 1 (i.e., it starts *Collecting*).

- *Cleaning:* robots (if any) on the left and bottom borders move leaving those nodes empty (it is performed only by robots in state 0).
- *Collecting:* robots move toward the left-upper corner of the grid (it is performed by robots in states 1 and 2).
- *Deploying:* robots follow a distinguished path on the grid, called *snake-path*, eventually occupying their final positions (it is performed by robots in states 3 and 4).

Note that due to asynchrony, at any point in time robots could be performing actions belonging to different phases. The asynchronous execution of the various phases requires special care in order to insure that robots continue to progress in their phase avoiding collisions and the creation of deadlocks.

Cleaning

The *Cleaning* phase is performed only by robots in state 0 and the goal is to have the robots move from the left and bottom borders toward the interior of the grid. This is done by having robots on the left border move down (if the down node is empty) and robots on the bottom border move up (if the top node is empty) or right (if the top node is occupied but the right node is empty). When moving up from the bottom border, a sensor enters state 1 and starts the *Collecting* phase.

Collecting

The *Collecting* phase is performed by robots in states 1 or 2 and the goal of the robots is to arrive to the left-upper corner. To avoid conflicts with robots possibly already in the *Deploying* phase, during the *Collecting* phase a sensor should not consider robots that are on the left border. The general rule for a sensor in the Collecting phase is to go up toward the upper-left corner (i.e., highest priority is given to movements up). However, depending on the neighboring conditions, to guarantee progress and avoid deadlocks, robots might have to move also left or right as described below.

Whenever a sensor (in state 1 or 2) has an empty upper node it goes *up* and stays in (or enters) state 1. If a sensor in state 1 is on the right neighbor of the upper left corner and the upper left corner is empty, it goes *left* and enters state 3 (*Deploying* phase). If a sensor in state 1 has all the visible nodes in its same column above itself occupied, and the left and bottom-left nodes are empty, it goes *left* and stays in state 1. Finally, a sensor (in state 1 or 2) goes *right* and stays in (or enters) state 2 if within its visibility radius all the following conditions are satisfied: all the nodes neither below nor right are occupied, the lower right node is empty, all the nodes in its same row on the right are empty, at least one of the above and right nodes is empty, and none of the robots in the above and right nodes at distance strictly less than $2d$ can go up. In all other cases a sensor stays in (or enters) state 1.

The movements of robots in the Collecting phase allow them to accumulate in a convenient shape around the upper left corner.

Deploying

This phase is executed by robots in states 3 and 4. A sensor starts this phase when it arrives to the upper left corner and enters state 3. In this phase the robots move on the snake-path (see Fig. 21.7) and eventually stop in their final positions, that is, the nodes $(i \cdot d, j \cdot d)$, with $0 \leq i, j \leq k$, called *final nodes*.

Let k be odd; the snake-path is the path $n_0, n_1, \ldots n_{(K-1)d}$, that starts at $n_0 = (0, n)$, ends at $n_{(K-1)d} = (d, n)$, and passes through every final node as shown in Fig. 21.7. By a slight modification of this path and, consequently, of the algorithm, the snake-path can be defined for k even.

Because of asynchrony, complications may arise. For example, if a sensor in the *Deploying* phase enters in contact with robots that are still in the *Collecting* phase,

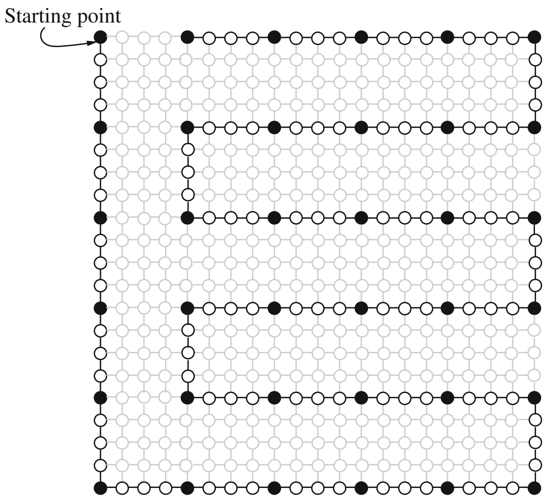


Fig. 21.7 The snake-path for $N = 21^2$, $K = 36$, $k = 5$, and $d = 4$

the sensor has to wait before continuing on the snake-path. Special care has to be taken and it can be shown that deadlocks are avoided and progress is guaranteed.

A sensor s enters state 3 when it reaches the left-upper corner. In this state, s follows the left border moving only if it sees above another sensor at distance lower than d and if the lower node is empty. If the sensor is in the left-upper corner it only moves if the right node is occupied, insuring in this way that there is at least one node in the *Collecting* phase. When the last node enters the *Deploying* phase it will stay at the upper-left corner and eventually the other nodes will position themselves at distance d from each other. When in state 3, a sensor s is following the left border of the grid. It enters state 4 when it reaches the bottom-left corner. A sensor in state 4 follows the snake-path from the left-bottom corner. A sensor s in the *Deploying* phase might see in the rows above some robots still performing the *Collecting* phase (the presence of these robots can be detected because of their "wrong" positions). In this case s does not move. If s does not see any sensor out of the snake-path but there are no robots in the $d - 1$ preceding nodes or the next node is occupied, then it waits. Otherwise s moves forward on the snake-path. When the last node enters the *Deploying* phase it will stay at the upper-left corner and eventually all other nodes will stop at their final position.

The correctness of the algorithm is proven by showing that no deadlocks can occur and that progress is guaranteed. Notice that although described for a square grid, the protocol works in any rectangular grid, provided the direction of the largest dimension is known. Since protocol SCATTER terminates within finite time with a uniform scattering, we have the following theorem.

Theorem 6 *The uniform deployment in rectangular spaces $\mathcal{G}$ can be solved without any collisions by sensors each having a constant amount of memory and a discrete visibility radius* $2d$ *in the* ASYNC *model.*

21.3.5 Incremental Deployment and Filling

21.3.5.1 Incremental Deployment as Filling

The task of uniform self-deployment is usually performed *after* the sensors have entered the space $\mathbb{U}$. This is because typically the sensors enter the space, all at once or in groups, without attention to the desired final placement criterion,[3] in a process called *injection*. Indeed, this is true in many situations especially in the case of very simple sensors. However, the separation between injection and self-deployment does not always occur. In fact, there are applications where the sensing entities are injected into the system one at a time, from one or few entry points; this is particularly true in the case of more complex (and/or delicate) sensorial entities (e.g., to avoid damage). In these situations, instead of having two separate processes, injection and self-deployment, the focus is on achieving the final goal directly, in a single process, called *incremental deployment*.

Howard et al. [45] proposed an incremental deployment algorithm for mobile robotic networks. Under the assumptions of global coordination, location awareness, and nodal visibility, that algorithm deploys robots one-at-a-time from a single entry point (*door*) and maintains a line of sight relationship between robots. They assume that every sensor is equipped with an ideal localization sensor; however, localization is very hard, especially for small sensors, so it cannot be generally assumed that the sensors know where they are. A very important and interesting mechanism they use is a logical orthogonal grid, superimposed on the space, that divides the space into cells, transforming the continuous space into an orthogonal cellular space. Let us stress that orthogonal spaces are interesting of their own, because they can be used to model indoor and urban environment; furthermore, the discretization of a continuous space into a cellular space is a rather common process used in a variety of contexts.

This approach has the additional advantage of reducing the problem of *incremental deployment* of an unknown arbitrary space $\mathbb{U}$ to the problem of *filling* an unknown cellular space. In the *Filling* problem, the mobile entities have to occupy all the cells of an unknown cellular space, entering through one or more designated entry points called *doors*; within finite time, the entities must reach a quiescent state, with exactly one entity in each cell. If two sensors are in the same cell at the same time then there is a *collision*. The algorithm executed by the sensors should avoid collisions (e.g., to prevent damage to the sensor or its sensory equipment).

The reduction to the filling problem is obtained by superimposing on the space $\mathbb{U}$ a logical *orthogonal grid* of the appropriate size; this will divide the space into cells (boundary cells might not be all within $\mathbb{U}$). Notice that the resulting cellular space $\mathcal{M}$ is orthogonal, i.e., polygonal with sides either parallel or perpendicular to one another (e.g., see Fig. 21.8). The space can be completed to become a bicolored

[3] For example, in some applications, sensors are dropped from the air.

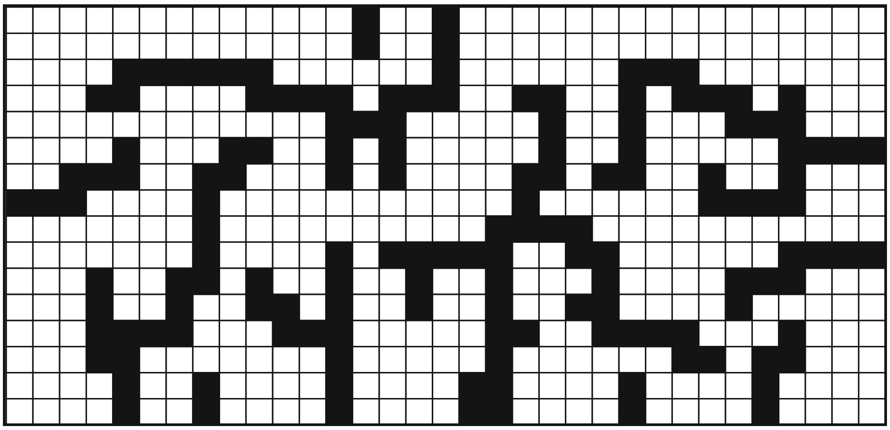


Fig. 21.8 A orthogonal space $\mathcal{M}$ (*white cells*) to be filled by the sensors and its enclosing cellular rectangle A

cellular rectangle[4] A, where each cell, called *pixel*, is colored *white* if it is part of $\mathcal{M}$, otherwise *black* (see Fig. 21.8).

At this point, to achieve an *incremental deployment* in the unknown arbitrary space $\mathbb{U}$ it is sufficient to perform a *filling* of the unknown orthogonal space $\mathcal{M}$; that is, filling the white pixels of A.

The problem of *filling* unknown orthogonal space $\mathcal{M}$ has been investigated by Hsiang et al. [47] for mobile sensors, and by Das et al., [16] for robotic networks.

In the study of Hsiang et al. [47], the sensors enter $\mathcal{M}$ from one or more doors. Their results are based on an ingenious follow-the-leader technique where each sensor communicates with the one following it and instructions to move are commu nicated from predecessor to successor. The sensors are anonymous but they need some persistent memory to remember whether or not is a leader and the direction of its movement. Since the algorithm uses only $O(1)$ bits of working memory in total, computationally the sensors can be just *finite-state machines*. In addition to requiring explicit communication, the solution of [47] assumes that the sensors operate in the SYNC model, which allows perfect coordination and synchronization between the sensors.

21.3.5.2 Filling by Robotic Sensors

For *robotic sensors* (where no direct communication exists), the filling problem of orthogonal spaces (and thus the incremental deployment problem) has been investigated by Das et al. [16]. First of all they proved that the sensors must have some *persistent* memory of the past for solving the filling problem successfully.

[4] A is the smallest cellular rectangle enclosing $\mathcal{M}$

Theorem 7 *The filling problem cannot be solved by oblivious sensors, even if they have unbounded visibility. This result holds even if there is only a single door and the model is* SSYNC.

Thus, some persistent memory is required. Indeed with just a constant amount of persistent memory, filling can be done in the case of a single door. This result is obtained in [16] with visibility radius of one in the SSYNC model.

Let the bicolored cellular rectangle A containing $\mathcal{M}$ be formed of pixels $p_{i,j}$, $1 \leq i \leq l$, $1 \leq j \leq c$; let *discrete* visibility radius of 1 mean that the sensor sees all eight neighboring cells (i.e., $v \geq \sqrt{2}q$, where q is the cell length). To understand the protocol, the structure of $\mathcal{M}$ will be represented by a graph $G = (V, E)$ defined as follows: First, partition each column into segments of consecutive white pixels ended by a black pixel in both extremes and numbered from top to down. Each segment is a node of G. Denote by $v_j^k \in V$ the node corresponding to the kth segment of column j, and by d_j^k the bottommost pixel of the segment v_j^k. There is an edge $\left(v_j^k, v_{j'}^{k'}\right) \in E$ if and only if (a) $j = j' + 1$ or $j = j' - 1$ and (b) there is a pixel $p_{i,j'} \in v_{j'}^{k'}$ neighbor to d_j^k or there is a pixel $p_{i,j} \in v_j^k$ neighbor to $d_{j'}^{k'}$.

It is easy to verify that the graph G so obtained is an acyclic connected graph (i.e., a tree). If there is an edge $\left(v_j^k, v_{j'}^{k'}\right)$ such that the bottommost pixel $d_j^k = p_{i,j}$ of v_j^k is a neighbor of the pixel $p_{i,j'} \in, v_{j'}^{k'}$, we say that $p_{i,j}$ is the ***entry point*** from v_j^k to $v_{j'}^{k'}$ and $p_{i,j'}$ is the ***entry point*** from $v_{j'}^{k'}$ to v_j^k.

The idea of the algorithm (FILLING — SINGLE DOOR) is to move the robots along the paths in G, starting from the node containing the door. Since the sensor can see the eight neighboring pixels, it can determine when it has reached an *entry point*. Let
$block^{+}(p_{i,j}) \equiv (p_{i,j}$ is empty $) \wedge ((p_{i-1,j}$ is **black**$) \vee (p_{i-1,j+1}$ is **black**$)))$
and let
$block^{-}(p_{i,j}) \equiv (p_{i,j}$ is empty $) \wedge ((p_{i-1,j}$ is **black**$) \vee (p_{i-1,j-1}$ is **black**$)))$.

```
Protocol FILLING - SINGLE DOOR
Meta-Rule: A sensor never backtracks.
Sensor $s$ in pixel $p_{i,j}$:
  if ( $p_{i+1,j}$ is empty ) then
    $s$ moves to $p_{i+1,j}$.
  else if ( $p_{i-1,j}$ is empty ) then
    $s$ moves to $p_{i-1,j}$.
  else if ( ( $block^{-}(p_{i,j-1})$ ) ) then
    $s$ moves to $p_{i,j-1}$.
  else if ( ( $block^{-}(p_{i,j+1})$ ) ) then
    $s$ moves to $p_{i,j+1}$.
  else
    $s$ does not move.
  end if
```

Since algorithm Single Door is collision free, it terminates in finite time and completely fills the space [16], we have the following theorem.

Theorem 8 *The filling problem for any space orthogonal space $\mathcal{M}$ with a single door can be solved without any collisions by sensors each having a constant amount of memory and a discrete visibility radius* 1 *in the* SSync *model.*

In the case of multiple doors, there are other strong limitations: the sensors must have a discrete visibility radius of at least 2 and they should not be indistinguishable [16]. Thus, for the problem to be solvable at all, sensors entering the space from different doors must be distinguishable, i.e., have different colors, and each sensor must have discrete visibility radius of at least 2. Indeed, under this assumption, the problem can be solved. However, the algorithm in this case is more complex with respect to the one with only a single door.

The idea of the algorithm, presented in [16], is as follows. Sensors coming from different doors (i.e., sensors of different colors) follow distinct paths in G and these paths do not intersect. In other words, the algorithm ensures that the cells visited by sensors of color c_i are occupied by sensors of the same color (and never by sensors of any other color). To achieve this, a sensor before moving to a pixel $p_{i,j}$ needs to determine if this pixel was visited by sensors of another color; fortunately, this can be done. Hence

Theorem 9 *The filling problem for any orthogonal space $\mathcal{M}$ with multiple doors can be solved without any collisions by sensors (with distinct color for distinct doors) each having a constant amount of memory and a discrete visibility radius* 2 *in the* SSync *model.*

21.4 Pattern Formation

The *pattern formation* problem is one of the most important coordination problem for robotic systems. The geometric pattern to be formed is a set of points (given by their Cartesian coordinates) in the plane, initially known by the entities. Initially the entities are in arbitrary positions, with the only requirement that no two entities are in the same position, and that, of course, the number of points prescribed in the pattern and the number of entities are the same. The robots are said to *form the pattern* if, at the end of the computation, the positions of the robots coincide, in everybody's local view, with the points of the pattern. Depending on the application, the formed pattern may be *translated*, and/or *rotated*, and/or *scaled*, and/or *flipped* into its mirror position with respect to the initial pattern. In particular, the pattern formation problem is said to be *scale free* if the formed pattern can be an arbitrarily scaled version of the input pattern.

The pattern formation problem is practically relevant because, if the robots can form a given pattern, they can agree on their respective roles in a subsequent, coordinated action. For this reason, it has been extensively investigated in the literature on autonomous robots (thus, without using communication), e.g., see

[3, 9, 19, 20, 35, 51, 53, 81, 83, 85, 89]. It has also been studied in systems using direct wireless communication (e.g., [37, 40, 56]).

The basic research questions are which patterns can be formed, and how they can be formed. In this section, we review the existing results on pattern formation by *mobile robotic sensors*, that is, by mobile entities that are silent and myopic. The spatial universe $\mathbb{U}$ is assumed to be the 2D space, and it is assumed that initially the sensors are arbitrarily dispersed in $\mathbb{U}$ but the visibility graph is connected.

21.4.1 Forming Scale-Free Patterns

Almost all protocols for pattern formation of silent autonomous robots assume *unlimited visibility*; in particular, they use the fact that each sensor can see all the other robots (e.g., [3, 9, 19, 20, 35, 51, 53, 81, 83, 85, 89, 90]). Thus, these protocols cannot be employed directly by robotic sensors, which by definition have a limited sensing range.

However, those same algorithms can be effectively used if the formed pattern can be an arbitrarily scaled version of the input pattern, i.e., for the *scale-free* pattern formation problem. This can be achieved by the following two-step strategy:

GATHER & FORM

1. Every sensors gets within the visibility range of every other sensor.
2. The sensors execute the relevant pattern formation protocol that assumes unlimited visibility.

The first step of this strategy requires solving the problem called *Near Gathering* [32]: starting from an initial arbitrary distribution in $\mathbb{U}$, the sensors, avoiding any collision, must within finite time reach a static equilibrium in which they are all mutually visible and on distinct locations; that is, there exists a time t when all sensors are in a state of static equilibrium, and for any two sensors s and r, $0 < |s(t) - r(t)| \leq v$. This problem is closely related to the *Rendezvous* or *Gathering* problem that will be discussed in detail in Sect. 21.5.

Once the first step has been performed, then the appropriate unlimited visibility pattern formation protocol can be started. In particular, the algorithms for *arbitrary pattern formation* (e.g., [35, 83, 90]) can be used by the sensors to form any input pattern. There are some provisos. In particular, to start the second step, a sensor must know that the execution of the first step has been completed; that is, all sensors are within its visibility range. However this implies that the number n of sensors must be known to the sensors. Another important point is that global mutual visibility, once reached in the first step, must be maintained throughout the execution of the second step. This necessary condition might not be of trivial enforcement; for example, there are some pattern formation algorithms that require, during their execution, some entities to move away from the others at a distance that (because of the limited visibility range of the sensors) might bring them out of the range of some other sensors.

In other words, the execution of the unlimited visibility pattern formation protocol must be carefully planned, tailored to the requirements of the pattern being formed, taking into account the movements of the sensors required by the algorithm: a non-trivial task. To date, no generic protocol is available for scale-free pattern formation by robotic sensors.

21.4.2 Circle Formation

A particular pattern extensively studied in the literature is the *circle*: the sensors, starting from arbitrary positions in the plane, have to arrange themselves in a circle of a given diameter D. Observe that this pattern formation problem is not scale free, and thus requires the agreement of the robots on the same unit distance.

If the sensors must be arranged at regular intervals on the boundary of a circle the problem is also called *uniform circle formation*. This kind of formation can be usefully deployed in surveillance tasks: the sensors are placed on the border of the area (or around the target) to surveil.

One of the first discussion on circle formation by a group of mobile entities was by Debest [17], who introduced it as an illustration of self-stabilizing distributed algorithms. He discussed the problem, but did not provide an algorithm.

The uniform circle formation problem was first studied by Sugihara and Suzuki [81]. They presented an heuristic that allowed the sensors to form an approximation of a circle having a given diameter $D \leq v$; it works without requiring common coordinate systems, the sensors can be oblivious, and the scheduling ASYNC. For sensor s, let $s_f(t)$ and $s_c(t)$ denote the position of the farthest and of the closest sensors at time t, respectively; let $\varepsilon > 0$ be an arbitrarily small predefined quantity. The protocol is rather simple:

```
Protocol CIRCLE CONVERGENCE (for sensor $s$ at time $t$)

1. If $|s_f(t) - s(t)| > 2D$ then move toward $s_f(t)$.
2. If $|s_f(t) - s(t)| < 2D - \varepsilon$, then move away from $s_f(t)$.
3. If $2D - \varepsilon \leq |s_f(t) - s(t)| \leq 2D$, then move away from $s_c(t)$.
```

Experiments have shown that sometimes the sensors converge toward a configuration similar to a *Reuleaux triangle* rather than a circle. Successively, the protocol has been improved by Tanaka [85], which proposed a new solution that produces a better approximation of the circle.

For the simpler SYNC model, a protocol that allows oblivious sensors without common coordinate system to converge toward a uniform placement on a circle has been recently proposed by Lee et al. as part of their investigation on forming concentric circles [55].

There are many other protocols for uniform circle formation [9, 19, 20, 22, 23, 53, 77]. For instance, the problem has been studied in the SSYNC setting by Défago and Konagaya [19], with anonymous and oblivious sensors, by presenting a solution that is a composition of two independent algorithms whereby the sensors first deterministically form a circle, and then converge to a situation in which all sensors are arranged uniformly on its boundary; simulation results of these studies have been presented in [77]. The solution in [19] is, however, computationally expensive: in fact, it involves the use of Voronoi diagrams, necessary to avoid the very specific possibility in which at least two robots share at some time the same position and also have total agreement on the coordinate system. Based on this observation, in [9] it is presented a new algorithm that avoids these expensive calculations; unfortunately, their solution relies on the simplifying assumption that sensors must not be located on the same radius, that radically changes the difficulty of the problem. Katreniak in [53], employing anonymous and oblivious sensors with no common coordinate system, handles to task of forming a *biangular circle* when the number of sensors is even: the sensors place themselves on the rim of a circle, each pair of adjacent sensors on the rim of the circle form with its center either an angle α or an angle β, and the angles alternate; the sensors act under ASYNC scheduling. When the number of sensors is odd, the sensors achieve the uniform circle. A solution that does not work for any number of robots has also been presented in [23], where the proposed oblivious algorithm works for a prime number of sensors in the semi-synchronous model. Dieudonné et al. [22] build upon the work of Katreniak [53] and extend it for the case with an even number of sensors; the algorithm solves the problem in finite time for any number n of sensors, except when $n = 4, 6$, and 8, under the SSYNC schedule. Also, here the sensors have the ability to reach exactly in one step their computed destination and cannot stop on the way. This assumption was later dropped in [24]; however, the algorithm in [24] still does not work for $n = 4$. Successively, Défago and Souissi presented in [20] an algorithm by which sensors deterministically form a circle in a finite number of steps and then asymptotically converge toward a situation in which they are positioned at regular intervals on the boundary of this circle, again under the SSYNC schedule. In contrast with the analogous two-part solutions previously presented in [19], here the two parts are combined in a single and simpler algorithm. However, even if this solution works for any number of sensors, it only converges to the uniform circle formation, in contrast with the solution in [24].

There are two major problems with all these solutions [9, 19, 20, 22, 23, 53, 77]. The first problem is that these protocols assume *unlimited visibility*; hence they can not be used directly by robotic sensors, by definition myopic. This means that first of all, robotic sensors can use these protocols only if the sensors are all mutually visible (i.e., the visibility graph is complete) and continue to be so throughout the execution of the protocol. The second problem is that these protocols are for *scale-free* circle formation; however, the problem we are facing has a fixed scale (given by the diameter D).

To overcome the first problem, the sensors can obtain an initial global mutually visibility by first performing a *Near Gathering* [32] and then execute the protocol (i.e., using the strategy GATHER & FORM discussed in Sect. 21.4.1); the difficulty of ensuring maintenance of global mutual visibility during execution clearly depends on which of those protocols is being used. To overcome the second problem, once a uniform circle formation has been obtained, an additional step is needed to *scale* the circle to the required dimension; this is not difficult provided $v \geq D$.

The problem of arranging robotic sensors in circular shapes has been studied by Miyamae et al., [68], considering robotic sensors whose vision is not only limited (i.e., within the visibility range v) but also *directional*. In fact, the vision function of each sensor detects another sensor within distance v with the center of the sensor assumed to be the origin and the direction of movement the reference angle (0°); however, the detection occurs only within three areas: forward (FV), and its left (LV) right (RV) sides; the backward area is that not detecting a sensor (see Fig. 21.9),

Furthermore, the sensors can only detect the presence of other sensors within their visibility areas, and not the exact number of sensors in their surrounding. In particular, each sensor can distinguish two scenarios for the forward area: zero sensors (FV= 0) or $\geq$ 1 sensors (FV= 1); for the left area, each sensor can distinguish three scenarios: zero sensor (LV= 00), one sensor (LV= 01), or more than one sensor (RV= 10); symmetrically, three scenarios can be detected for the right area as well. Based on this simple information, each sensor acts as described in the following protocol.

Protocol EMERGENT CIRCLE (for sensor s_i)

1. If FV= 0, LV= 01, and RV= 00, then turn left.
2. If FV= 0, LV= 00, and RV= 01, then turn right.
3. If FV= 0, LV= 01, and RV= 01, then turn to the last previous direction.
4. For the others scenarios, proceed straight.

The emergent behavior of the sensors following these simple rules has been analyzed by computer simulations in [68]. The experimental results show that the formation of the circle depends on the number of sensors and the front and side view angles of local vision, demonstrating that the front view angle must be between 15°

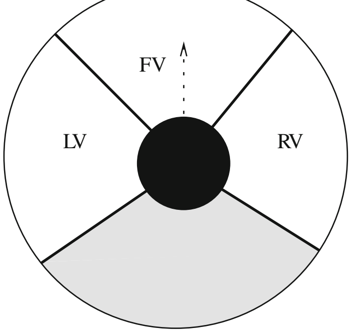


Fig. 21.9 The vision model for the emergent approach to circle formation. The *black circle* represents the robots; the *dark area* is the blind zone of the robot

and 75°, while the side view angles between 60° and 120°. Another interesting observation arising from the simulations is that the circle formation rate decreases the larger the number of sensors is; that is, an excessive number of sensors affects negatively the formation process.

21.5 Gathering

In systems of mobile entities, one of the most basic coordination and synchronization task is that of *gathering*: the entities, placed in arbitrary positions in $\mathbb{U}$, must congregate at a single location (the choice of the location is not predetermined). This fundamental problem is also called *rendezvous* or *homing*. If the entities are seen as points, the gathering problem is the one of having all entities move to the same point, that is, forming the special pattern *point*; hence the problem is sometimes called *point formation*.

The gathering problem has been extensively investigated both experimentally and theoretically in the *unlimited visibility* setting, that is, assuming that the entities are capable to sense ("see") the entire space (e.g., see [1, 10, 12, 18, 35, 49, 81, 83]).

In general, and more realistically, sensors can sense only a surrounding within a radius of bounded size (refer to the example depicted in Fig. 21.1). This setting, which is the one in which robotic sensors operate, is understandably more difficult; for example, a sensor might not even know the total number of sensors nor where they are located if outside its radius of visibility. Not surprisingly, not many algorithmic results are known (e.g., [2, 3, 34, 59, 60, 79]). They are reviewed in this section according to the scheduling model assumed: ASYNC, SSYNC, and SYNC.

21.5.1 Asynchronous Gathering

The most difficult setting for the gathering problem is clearly the *asynchronous* one, where little or no timing assumptions are made. In the literature, there are only few algorithms tackling asynchrony when gathering robotic sensors [34, 60].

In the investigation of Lin et al. [60], a limited form of asynchrony is considered where the time required by the *Wait*, *Locate*, and *Compute* states is bounded by a globally predefined amount, while the time spent in the *Move* state by sensor i is bounded by a locally predefined quantity (i.e., not necessarily the same for each sensor). This form of asynchrony lies in between the ASYNC model and the SSYNC model. The resulting solution allows a set of non-oblivious sensors with limited visibility to *converge* toward a single point.

The fully asynchronous model ASYNC is considered only in the investigation of Flocchini et al. [34]. They show that the availability of *orientation*[5] allows a set of anonymous oblivious sensors with limited visibility to *gather* at a single

[5] i.e., agreement on axes and directions (positive vs. negative) of a common coordinate system, but not necessarily on the origin nor on the unit distance.

point in finite time. This result holds not only allowing each activity and inactivity of the sensors to be totally unpredictable (but finite) in duration, but also making their movement toward a destination unpredictable (but not infinitesimally small) in length. In the rest of this section, we look at this result in more detail.

Let *Right* be the rightmost vertical axis where some sensor initially lie. The idea of the algorithm is to make the sensors move toward *Right*, in such a way that, after a finite number of steps, they will reach it and gather at the bottommost position occupied by a sensor at that time.

Let s perform a *Locate* operation at time t; as a result, it has available its circle of visibility $\mathcal{C}_t(s)$ with the positions of all the sensors in it at time t. The algorithm describes the computation that s will now do with this input. Different destination points will be computed depending on the positions of the sensors in its circle of visibility; once the computation is completed, s starts moving toward its destination (but it may stop before the destination is reached). Informally,

- If s sees sensors to its left or above on its vertical axis, it does not move.
- If s sees sensors only below on its vertical axis, it moves down toward the nearest sensor.
- If s sees sensors only to its right, it moves horizontally toward the vertical axis of the nearest sensor.
- If s sees sensors both below on its axis and on its right, it computes a destination point and performs a diagonal move to the right and down, as explained below.

To describe the diagonal movement in detail we need to introduce some notation (refer to Fig. 21.10). Let $\overline{AA'}$ be the vertical diameter of $\mathcal{C}_t(s)$ with A' as the top end point and A the bottom end point; let $\mathcal{R}_s$ denote the topologically open region (with respect to $\overline{AA'}$) inside $\mathcal{C}_t(s)$ and to the right of s and let $S = \overline{sA}$ and $S' = \overline{sA'}$, where both S' and S are topologically open on the s side (i.e., s belongs neither to S' nor to S). Let Ψ be the vertical axis of the sensor in $\mathcal{R}_s$, if any, nearest to s with respect to its projection on the horizontal axis. We are now ready to describe the details of the diagonal movement routine:

```
Diagonal_Movement($\Psi$)
    $B$ := upper intersection between $\mathcal{C}_t(s)$ and $\Psi$;
    $C$ := lower intersection between $\mathcal{C}_t(s)$ and $\Psi$;
    $A$ := point on $S$ at distance $v$ from $s$;
    $2\beta = A\widehat{s}B$;
    **if** $\beta < 60°$ **then**
        $(B, \Psi)$ := Rotate($s$, $B$);
    **end if**
    **H** := Diagonal_Destination($\Psi$, $A$, $B$);
    Move(**H**).
```

where Rotate() and Diagonal_Destination() are as follows:

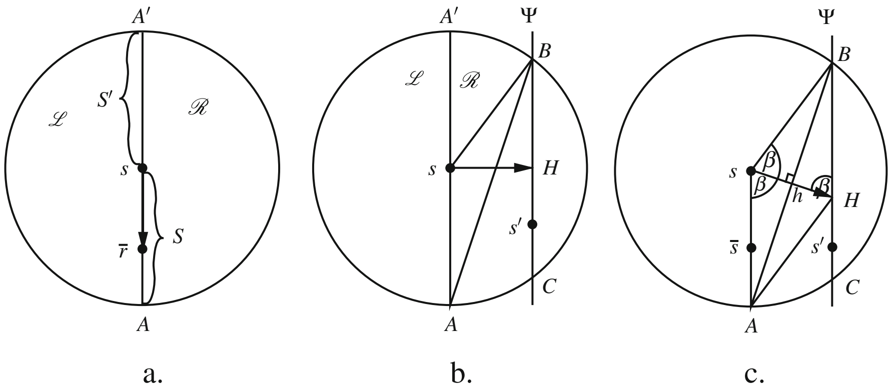


Fig. 21.10 (**a**) Notation used in the Gathering algorithm; (**b**) horizontal move; and (**c**) diagonal move

– `Rotate(`s, B`)` rotates the segment $\overline{sB}$ in such a way that $\beta = 60°$ and returns the new position of B and Ψ. This choice of angle ensures that the destination point is not outside the circle (see Fig. 21.11).
– `Diagonal_Destination(`Ψ, A, B`)` computes the destination of s in the following way: the direction of s's movement is given by the perpendicular to the segment $\overline{AB}$; the destination of s is the point H on the intersection of the direction of its movement and of the axis Ψ.

The correctness of the algorithm is proven by first showing that the sensors which are initially visible will stay visible until the end of the computation, and then that the robots' movement leads to non-infinitesimally small progress toward gathering thus concluding that all sensors will gather in a point on *Right* in finite time. We then have the following theorem.

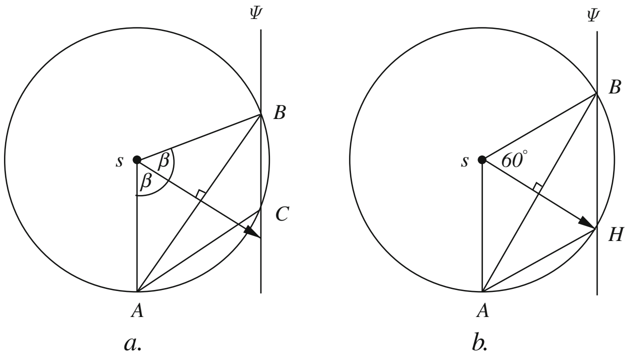


Fig. 21.11 Routine `Rotate()`: in (**a**), $\beta < 60°$; in (**b**) the scenario after `Rotate()` has been executed

Theorem 10 *In the* Async *model a set of anonymous oblivious sensors in* $\mathbb{R}^2$*, endowed with orientation, can* gather *at a single point in finite time.*

Notice that the proposed algorithm does *not* assume that the sensors have the capability of *multiplicity detection* (i.e., the ability to determine in the sensing phase if more than one sensor is in a given location).

As mentioned, the above algorithm requires an agreement on the coordinate system. The problem of creating such an agreement in Async has been studied by Samiloglu, Gazi, and Bugra Koku, who have proposed several strategies that have been experimentally observed to converge toward a common orientation [78].

Probabilistic protocols for gathering in the absence of agreement on the coordinate systems have been proposed and experimentally analyzed by Soysal et al. [80].

21.5.2 Semi-Synchronous Gathering

In the Ssync model, the gathering problem has also been tackled by Ando et al. [2]. In contrast with the setting considered in Sect. 21.5.1, here the sensors do not have any kind of common orientation. However, the compass has been traded with the semi-synchronicity of the sensors. Moreover, with the solution of [2], the robots only *converge* toward a gathering point.

Let $P(t) = \{s_1(t), \dots, s_n(t)\}$ denote the set of the n sensors' positions at time t. Also, let $S_i(t)$ denote the set of sensors that are within distance v from s_i at time t; that is, the set of sensors that are visible from s_i (note that $s_i \in S_i(t)$). $SC_i(t)$ denotes the smallest enclosing circle of the set $\{s_j(t) | s_j \in S_i(t)\}$ of the positions of the sensors in $S_i(t)$ at t; let $c_i(t)$ be the center of $SC_i(t)$.

The algorithm is described below (refer also to Fig. 21.12).

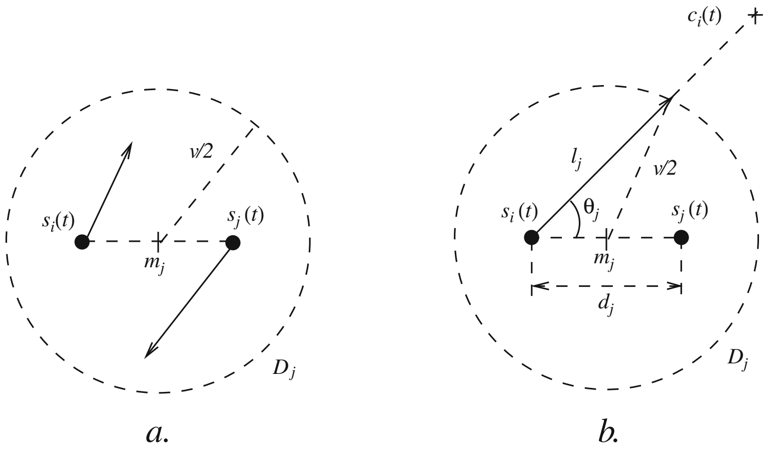


Fig. 21.12 The algorithm for the gathering problem in Ssync

```
SEMI-SYNCH GATHERING ALGORITHM
1. If $S_i(t) = \{s_i\}$, then $x = s_i(t)$.
2. $\forall s_j \in S_i(t) \setminus \{s_i\}$,
    2.1. $d_j = dist(s_i(t), s_j(t))$,
    2.2. $\theta_j = c_i(t)\widehat{s_i(t)}s_j(t)$,
    2.3. $l_j = (d_j/2)\cos\theta_j + \sqrt{(v/2)^2 - ((d_j/2)\sin\theta_j)^2}$,
3. $LIMIT = \min_{s_j \in S_i(t)\setminus\{s_i\}}\{l_j\}$,
4. $GOAL = dist(s_i(t), c_i(t))$,
5. $MOVE = \min\{GOAL, LIMIT, \sigma\}$,
6. $x =$ point on $[s_i(t)c_i(t)]$ at distance $MOVE$ from $s_i(t)$.
```

Every time a sensor s_i becomes active, it moves toward $c_i(t)$, but only over a certain distance $MOVE$. Specifically, if s_i does not see any sensor other than itself, then s_i does not move at all. Otherwise, the algorithm chooses as next position for s_i the point x on the segment $\overline{s_i(t)c_i(t)}$ that is closest to $c_i(t)$ and that satisfies the following conditions:

1. $dist(s_i(t), x) \leq \sigma$. Note that this means that the sensors agree on an arbitrary small constant $\sigma > 0$, a priori known, and they use it to bound the distance traveled by a sensor in one step.
2. For every sensor $s_j \in S_i(t)$, x lies in the disk D_j whose center is the midpoint m_j of $s_i(t)$ and $s_j(t)$ and whose radius is $v/2$. This condition ensures that s_i and s_j will still be visible after the movement of s_i (and possibly of s_j, see Fig. 21.12.a).

The correctness proof is based on the following reasoning: First, two sensors that are connected in the visibility graph at time t will stay connected at time $t+1$. In fact, if $s_i(t)$ and $s_j(t)$ are connected, then $s_j(t) \in S_i(t)$ and $s_i(t) \in S_j(t)$ and then, by definition of $LIMIT$, both $s_i(t+1)$ and $s_j(t+1)$ lie inside the disc with center m_j (Fig. 21.12a). Second, let $CH(t)$ be the convex hull of the sensors at time t, for any $t \geq t_0$, $CH(t+1) \subseteq CH(t)$ leading to the proof that $CH(t)$ converges to a point. Then we have

Theorem 11 *In the* SSYNC *model, a set of anonymous oblivious sensors in* $\mathbb{R}^2$ *can converge to a gathering point.*

The problem has also been examined in the same model SSYNC when there are inaccuracies or faults. In particular, Gordon, Wagner, and Bruckstein have investigated the case when the sensors cannot accurately measure the distance from their neighbors and hence cannot rely on this information [41, 42]. The gathering problem has also been examined with respect to the availability of compasses. As discussed in Sect. 21.5.1, the presence of reliable compasses allows the sensors to gather in finite time even in the ASYNC model (Theorem 10) (and thus also in the SSYNC model). The problem of gathering when compasses are unstable for some arbitrary long periods has been studied by Soussi et al. [79]. They proved that in the SSYNC model, the sensors will gather in finite time, provided that the compasses stabilize eventually.

21.5.3 Fully Synchronous Gathering

There have been several investigations on the gathering problem with sensors operating in the fully synchronous scenario, i.e., in model SYNC [39, 59, 64, 91]. The starting point of these investigations is the convergence protocol of Ando et al. [2], described in Sect. 21.5.2, operating in the SSYNC and thus in the SYNC models. Like [2], these protocols work for oblivious sensors with no common coordinate system, and they all converge toward a unique point; unlike [2], they are only for the SYNC model.

Lin et al. [59] propose and analyze a family of convergence algorithms for gathering in the plane based on [2]. In [64] variants of the general strategy described in [59] are developed: Martínez considers the presence of noisy measurements of neighbors [64].

In all the above investigations on gathering, as well as those in the ASYNC and SSYNC models discussed before, the universe $\mathbb{U}$ in which the gathering was taking place was (implicitly) assumed to be either the entire plane or a *convex* region of the plane.

The case when the sensors operate in a *non-convex* region (see Fig. 21.13), of which they have no map, has been considered only by Ganguli, Cortés, and Bullo in [39]. In such a space, two sensors s and s' are said to be *mutually visible* at time t if not only their distance is at most v but also the segment connecting their positions at time t is completely contained in $\mathbb{U}$;. for instance, sensors s and s' in Fig. 21.13 are within distance v, but they are not mutually visible. The approach used to solve the problem is that of computing a set of constraints that the sensors have to follow when moving so that (a) the mutual visibility graph stays connected during the movements and (b) the distances between sensor strictly decrease at each time step.

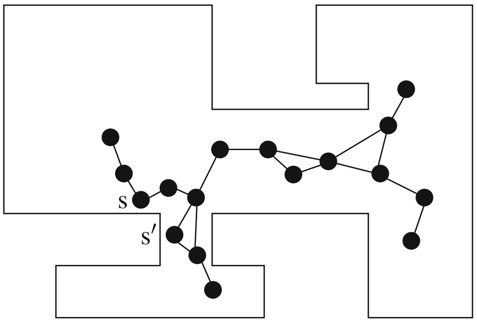


Fig. 21.13 An example of non-convex environment for the gathering problem in [39]. The edges between sensors represent the edges of the visibility graph

The first set of constraints is derived from those by Ando et al. [2] discussed in Sect. 21.5.2 and guarantees that condition (a) is met; in particular, it imposes that if two sensors s_i and s_j are mutually visible at time t, they stay connected at time $t+1$: let $p_i = s_i(t)$ and $p_j = s_j(t)$ be the positions of sensors s_i and s_j at time t, respectively, then s_i and s_j are allowed to move inside the ball B of radius $\frac{v}{2}$ centered in the midpoint of p_i and p_j (see Fig. 21.12). Clearly, since the sensors operate in a non-convex environment, the sensors are limited to move inside any convex area contained in the intersection between B and $\mathbb{U}$: in [39] the *Constraint Set Generator Algorithm* is given to allow the computation of such a convex area by any pair of mutually visible sensors.

The overall idea of the gathering algorithm, called the *Perimeter Minimizing Algorithm*, can be summarized as follows: at each time step t, each sensor computes all convex areas resulting from executing the *Constraint Set Generator Algorithm* for all its neighbors in the v-range visibility graph at t; the area where it is allowed to move in order to verify condition (a) is therefore the intersection of all these areas. The sensor moves now toward the circumcenter of its allowed moving zone, i.e., the center of the smallest circle enclosing this area. It can be proven that this choice satisfies condition (b) above.

Theorem 12 *In the* SYNC *model, a set of anonymous oblivious sensors in operating in a non-convex environment can converge to a gathering point.*

The behavior of this protocol has been analyzed experimentally in [39] also when (i) the sensors operate asynchronously or (ii) the sensors can introduce distance and direction errors in both sensing and moving, or (iii) the sensors are modeled as disks; the results give an indication of the robustness of the protocol with respect to those three factors.

21.5.4 Coalescence

An interesting problem related to the gathering is *Coalescence*: arbitrarily dispersed (and possibly isolated) mobile sensors must independently search for their fellow sensors with the goal of being all within a given distance. Note that if that distance is not greater than v, this is precisely the *Near Gathering* problem whose goal is to form a single connected visibility graph.

COALESCENCE

3. An isolated sensor
   a. uniformly chooses a direction of movement θ in $[0, 2\pi)$.
   b. moves following the chosen direction with a constant speed for a constant distance.
4. When two sensors "meet" (i.e., they are within distance v) they form a single cluster: they stay connected to each other and move together following the same (random) path.
5. When two clusters meet they coalesce to form a single cluster.

The coalescence problem has been investigated by Poduri and Sukhatme in [73]. They consider sensors performing an independent random search of the other sensors according to the following set of rules.

Given this set of rules, considering a fully synchronous scenario (i.e., the SYNC model), the main question addressed in [73] is how long will it take for the sensors to coalesce into a single connected visibility graph. Clearly, the *spread* of the clusters plays an important role: in fact, if the sensors in each cluster remain spread out, the disconnected sensors have higher probability of being discovered. That is, here the focus is on connectivity rather than on colocation. They show analytically that coalescence time has an exponential distribution which is a function of the number of sensors, spread, communication range, and size of the domain. Also, as the number of sensors increases, coalescence time decreases as $O\left(\frac{1}{\sqrt{n}}\right)$ and $\Omega\left(\frac{1}{n}\log n\right)$. Simulation experiments support the analytical results, suggesting that the lower bounds derived analytically for coalescence time is tight.

21.6 Conclusions and Open Problems

Several research questions are still open. With respect to *gathering*, the outstanding open problem is whether it is possible to gather when the robotic sensors are asynchronous, oblivious, and without common orientation.

For *self-deployment*, the foremost open problem is the determination of whether knowledge of d is indeed necessary for exact self-deployment in an *oriented ring*. Should this be the case, the research goal becomes to determine which is the "weakest" additional assumption (e.g., a priori knowledge, capability) that would make exact self-deployment possible. A more general and challenging open problem is to find additional sensors' capabilities that would enable the existence of an asynchronous exact self-deployment protocol in *unoriented rings*.

The impact that sensorial errors and inaccuracies have on the correctness of the algorithms should be studied in detail. New algorithms are needed for different assumptions on the visibility power of the sensors; for instance, the accuracy of the sensors' ability to detect the other sensors' positions might decrease with the distance. Another important concern is clearly the one of the presence of possible faulty sensors. The fault tolerant issues have been recently addressed in [1, 52, 79], but only in the unlimited visibility setting.

Finally, an open and important research direction is to identify meaningful efficiency parameters and study the computational complexity of the problem. In fact, in all existing investigations, the complexity of the solutions has never been an issue; indeed, there is an absence of cost measures.

Slightly faulty snapshots, obstacles that limit the visibility and that moving sensors must avoid or push aside, sensors that appear and disappear from the scene, as well as precise cost measures, clearly suggest that the algorithmic nature of distributed coordination of autonomous, mobile robotic sensors is far from been completed and further investigations are clearly needed.

Acknowledgments This work has been supported in part by the Natural Sciences and Engineering Research Council of Canada, under Discovery Grants, and by PRIN 2008—MadWeb.

References

1. N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM Journal on Computing*, 36:56–82, 2006.
2. H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. A distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, 1999.
3. H. Ando, I. Suzuki, and M. Yamashita. Formation and agreement problems for synchronous mobile robots with limited visibility. In: *Proceedings of IEEE Symposium of Intelligent Control*, pages 453–460, 1995.
4. L. Barrière, P. Flocchini, E. Mesa-Barrameda, and N. Santoro. Uniform scattering of autonomous mobile robots in a grid. *International Journal of Foundations of Computer Science*, to appear (2011).
5. M. A. Batalin and G. S. Sukhatme. Coverage, exploration and deployment by a mobile robot and communication network. *Telecommunication Systems*, 26(2):181–196, 2004.
6. D. Bhadauria and V. Isler. Data gathering tours for mobile robots. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3868–3873, 2009.
7. F. Bullo, J. Cortes, and S. Martinez. Distributed Control of Robotic Networks. Princeton University Press, Princeton, NJ, 2009.
8. Y. U. Cao, A. S. Fukunaga, A. B. Kahng, F. Meng. Cooperative mobile robotics: Antecedents and directions. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 226–234, 1995.
9. I. Chatzigiannakis, M. Markou, S. Nikoletseas. Distributed circle formation for anonymous oblivious robots. In: *Proceedings of 3rd International Workshop on Experimental and Efficient Algorithms (WEA)*, pages 159–174, 2004.
10. M. Cieliebak, P. Flocchini, G. Prencipe, N. Santoro. Solving the gathering problem. In: *Proceedings of 30th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1181–1196, 2003.
11. J. Clark, R. Fierro. Mobile robotic sensors for perimeter detection and tracking. *ISA Transactions*, 46(1):3–13, 2007.
12. R. Cohen, D. Peleg. Convergence properties of the gravitational algorithm in asynchronous robot systems. *SIAM Journal on Computing*, 34:1516–1528, 2005.
13. R. Cohen, D. Peleg. Local spreading algorithms for autonomous robot systems. *Theoretical Computer Science*, 399:71–82, 2008.
14. K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, G.S. Sukhatme. Robomote: enabling mobility in sensor networks. In: *Proceedings of 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 404–409, 2005.
15. S. Das, P. Flocchini, S. Kutten, A. Nayak, N. Santoro. Map construction of unknown graphs by multiple agents. *Theoretical Computer Science*, 385(1–3):34–48, 2007.
16. S. Das, E. Mesa-Barrameda, N. Santoro. Deployment of asynchronous robotic sensors in unknown orthogonal environments. In: *Proceedings of 4th International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSOR)*, pages 25–140, 2008.
17. X. A. Debest. Remark about self-stabilizing systems. *Communication of the ACM*, 2(38):115–177, 1995.
18. X. Défago, M. Gradinariu, S. Messika, P. R. Parvédy. Fault-tolerant and self-stabilizing mobile robots gathering. In: *Proceedings of 20th International Symposium on Distributed Computing (DISC)*, pages 46–60, 2006.

19. X. Défago, A. Konagaya. Circle formation for oblivious anonymous mobile robots with no common sense of orientation. In: *Proceedings of Workshop on Principles of Mobile Computing*, pages 97–104, 2002.
20. X. Défago, S. Souissi. Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. *Theoretical Computer Science*, 396(1–3):97–112, 2008.
21. Y. Dieudonné, S. Dolev, F. Petit, M. Sega. Deaf, dumb, and chatting asynchronous robots. In: *Proceedings of 13th International Conference on Principles of Distributed Systems (OPODIS)*, LNCS 5923, pages 71–85, 2009.
22. Y. Dieudonné, O. Labbani-Igbida, F. Petit. Circle formation of weak mobile robots. *ACM Transactions on Autonomous and Adaptive Systems*, 3(4):1–20, 2008.
23. Y. Dieudonné, F. Petit. Circle formation of weak robots and lyndon words. *Information Processing Letters*, 4(104):156–162, 2007.
24. Y. Dieudonné, F. Petit. Swing words to make circle formation quiescent. In: *Proceedings of 14th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, LNCS 4474, pages 166–179, 2007.
25. E. W. Dijkstra. *Selected Writings on Computing: A Personal Perspective*. Springer, New York, NY, 1982.
26. S. Dobrev, P. Flocchini, G. Prencipe, N. Santoro. Searching for a black hole in arbitrary networks: optimal mobile agents protocols. *Distributed Computing*, 19(1):1–19, 2006.
27. M. Dunbabin, P. Corke, I. Vasilescu, D. Rus. Data muling over underwater wireless sensor networks using an autonomous underwater vehicle. In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2006.
28. A. Efrima, D. Peleg. Distributed models and algorithms for mobile robot systems. In: *Proceedings of 33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, LNCS 4362, pages 70–87, 2007.
29. Y. Elor, A. M. Bruckstein. Multi-agent deployment and patrolling on a ring graph. Technical Report CIS-2009-16, Computer Science Department, Technion, Israel, 2009.
30. N. Fatès. Solving the decentralized gathering problem with a reaction-diffusion-chemotaxis scheme. Swarm Intelligence (to appear), 4(2): 91–115, 2010.
31. P. Flocchini, G. Prencipe, N. Santoro. Self-deployment algorithms for mobile sensors on a ring. *Theoretical Computer Science*, 402(1):67–80, 2008.
32. P. Flocchini, G. Prencipe, N. Santoro. Near gathering of weak robots with limited visibility: Algorithms and applications. Technical report, Carleton University, 2010.
33. P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer. Pattern formation by autonomous robots without chirality. In: *Proceedings of 8th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 147–162, 2001.
34. P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer. Gathering of asynchronous mobile robots with limited visibility. *Theoretical Computer Science*, 337:147–168, 2005.
35. P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer. Arbitrary pattern formation by asynchronous oblivious robots. *Theoretical Computer Science*, 407(1–3):412–447, 2008.
36. P. Fraigniaud, L. Gasieniec, D. Kowalski, A. Pelc. Collective tree exploration. *Networks*, 48:166–177, 2006.
37. J. Fredslund, M. J. Matarić. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, 2002.
38. A. Ganguli, J. Cortes, F. Bullo. Visibility-based multi-agent deployment in orthogonal environments. In: *Proceedings of American Control Conference*, pages 3426–3431, 2007.
39. A. Ganguli, J. Cortés, F. Bullo. Multirobot rendezvous with visibility sensors in nonconvex environments. *IEEE Transactions on Robotics*, 25(2):340–352, 2009.
40. S. Gilbert, N. Lynch, S. Mitra, T. Nolte. Self-stabilizing robot formations over unreliable networks. *ACM Transactions on Autonomous and Adaptive Systems*, 4(3):1–29, 2009.
41. N. Gordon, Y. Elor, A. M. Bruckstein. Gathering multiple robotic agents with crude distance sensing capabilities. In: *Proceedings of 6th International Conference on Ant Colony Optimization and Swarm Intelligence*, LNCS 5217, pages 72–83, 2008.

42. N. Gordon, I. A. Wagner, A. M. Bruckstein. Gathering multiple robotic a(ge)nts with limited sensing capabilities. In: *Proceedings of 2nd International Conference on Ant Colony Optimization and Swarm Intelligence*, LNCS 3172, pages 142–153, 2004.
43. N. Heo, P. K. Varshney. A distributed self spreading algorithm for mobile wireless sensor networks. In: *Proceedings of IEEE Wireless Communication and Networking Conference*, volume 3, pages 1597–1602, 2003.
44. N. Heo, P. K. Varshney. Energy-efficient deployment of intelligent mobile sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, 35(1):78–92, 2005.
45. A. Howard, M. J. Mataric, G. S. Sukhatme. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots*, 13(2):113–126, 2002.
46. A. Howard, M. J. Mataric, G. S. Sukhatme. Mobile sensor network deployment using potential fields. In: *Proceedings of 6th International Symposium on Distributed Autonomous Robotics Systems (DARS)*, pages 299–308, 2002.
47. T.-R. Hsiang, E. Arkin, M. A. Bender, S. Fekete, J. Mitchell. Algorithms for rapidly dispersing robot swarms in unknown environments. In: *Proceedings of 5th Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 77–94, 2002.
48. Y. Ikemoto, Y. Hasegawa, T. Fukuda, K. Matsuda. Gradual spatial pattern formation of homogeneous robot group. *Information Sciences*, 171(4):431–445, 2005.
49. D. Jung, G. Cheng, A. Zelinsky. Experiments in realising cooperation between autonomous mobile robots. In: *Proceedings of 5th International Symposium on Experimental Robotics (ISER)*, pages 513–524, 1997.
50. A. Kansal,W. Kaiser, G. Pottie, M. Srivastava, G. S. Sukhatme. Reconfiguration methods for mobile sensor networks. *ACM Transactions on Sensor Networks*, 3(4):22–23, 2007.
51. M. Kasuya, N. Ito, N. Inuzuka, K. Wada. A pattern formation algorithm for a set of autonomous distributed robots with agreement on orientation along one axis. *Systems and Computers in Japan*, 37(10):89–100, 2006.
52. Y. Katayama, Y. Tomida, H. Imazu, N. Inuzuka, K. Wada. Dynamic compass models and gathering algorithms for autonomous mobile robots. In: *Proceedings of 14th Colloquium on Structural Information and Communication Complexity (SIROCCO)*, LNCS 4474, 2007.
53. B. Katreniak. Biangular circle formation by asynchronous mobile robots. In: *Proceedings of 12th International Colloquium on Structural and Communication Complexity (SIROCCO)*, LNCS 3499, pages 185–199, 2005.
54. O. Kosut, A. Turovsky, J. Sun, M. Ezovski, G. Whipps, L. Tong. Integrated mobile and static sensing for target tracking. In: *Proceedings of Military Communications Conference (MILCOM)*, pages 1–7, 2007.
55. G. Lee, S. Yoon, N. Y. Chong, H. Christensen. A mobile sensor network forming concentric circles through local interaction and consensus building. *Journal of Robotics and Mechatronics*, 21(4):469–477, 2009.
56. J. Lee, S. Venkatesh, M. Kumar. Formation of a geometric pattern with a mobile wireless sensor network. *Journal of Robotic Systems*, 21(10):517–530, 2004.
57. X. Li, H. Frey, N. Santoro, I. Stojmenovic. Focused coverage by mobile sensor networks. In: *Proceedings of 6th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 466–475, 2009.
58. X. Li, N. Santoro. An integrated self-deployment and coverage maintenance scheme for mobile sensor networks. In: *Proceedings of 2nd International Conference on Mobile Ad-Hoc and Sensors Networks (MSN)*, pages 847–860, 2006.
59. J. Lin, A.S. Morse, B.D.O. Anderson. The multi-agent rendezvous problem. Part 1: The synchronous case. *SIAM Journal on Control and Optimization*, 46(6):2096–2119, 2007.
60. J. Lin, A.S. Morse, B.D.O. Anderson. The multi-agent rendezvous problem. Part 2: The asynchronous case. *SIAM Journal on Control and Optimization*, 46(6):2120–2147, 2007.
61. B. Liu, P. Brass, O. Dousse, P. Nain, D. Towsley. Mobility improves coverage of sensor networks. In: *Proceedings of 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 300–308, 2005.

62. L. Loo, E. Lin, M. Kam, P. Varshney. Cooperative multi-agent constellation formation under sensing and communication constraints. *Cooperative Control and Optimization*, pages 143–170, 2002.
63. N. Lynch, S. Mitra, T. Nolte. Motion coordination using virtual nodes. In: *Proceedings of 44th IEEE Conference on Decision and Control*, 2005.
64. S. Martínez. Practical multiagent rendezvous through modified circumcenter algorithms. *Automatica*, 45(9):2010–2017, 2009.
65. S. Martínez, F. Bullo, J. Cortes, E. Frazzoli. On synchronous robotic networks—parts i and ii. *IEEE Transactions on Automatic Control*, 52(12):2199–2226, 2007.
66. E. Martinson, D. Payton. Lattice formation in mobile autonomous sensor arrays. In: *Proceedings of International Workshop on Swarm Robotics (SAB)*, pages 98–111, 2004.
67. Y. Mei, C. Xian, S. Das, Y.C. Hu, Y.-H. Lu. Sensor replacement using mobile robots. *Computer Communications*, 30(13):2615–2626, 2007.
68. T. Miyamae, S. Ichikawa, F. Hara. Emergent approach to circle formation by multiple autonomous modular robots. *Journal of Robotics and Mechatronics*, 21(1):3–11, 2009.
69. S. E. Nikoletseas. Models and algorithms for wireless sensor networks (smart dust). In: *Proceedings of 32nd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, LNCS 3831, pages 64–83, 2006.
70. Y. Oasa, I. Suzuki, M. Yamashita. A robust distributed convergence algorithm for autonomous mobile robots. In: *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pages 287–292, 1997.
71. M.R. Pac, A. M. Erkmen, I. Erkmen. Scalable self-deployment of mobile sensor networks: A fluid dynamics approach. In: *Proceedings of IEEE/RSJ International Conference Intelligent Robots and Systems*, pages 1446–1451, 2006.
72. S. Poduri, G. S. Sukhatme. Constrained coverage for mobile sensor networks. In: *Proceedings of IEEE International Conference on Robotic and Automation*, pages 165–173, 2004.
73. S. Poduri, G. S. Sukhatme. Achieving connectivity through coalescence in mobile robot networks. In: *Proceedings of 1st ACM International Conference on Robot Communication and Coordination (RoboComm)*, volume 318, pages 1–6, 2007.
74. O. Powell, P. Leone, J. Rolim. Energy optimal data propagation in wireless sensor networks. *Journal of Parallel and Distributed Computing*, 67(3):302–317, 2007.
75. G. Prencipe. The effect of synchronicity on the behavior of autonomous mobile robots. *Theory of Computing Systems*, 38:539–558, 2005.
76. G. Prencipe, N. Santoro. Distributed algorithms for mobile robots. In: *Proceedings of 5th IFIP International Conference on Theoretical Computer Science (TCS)*, 2006.
77. S. Samia, X. D´efago, T. Katayama. Convergence of a uniform circle formation algorithm for distributed autonomous mobile robots. In: *Proceedings of Journés Scientifiques Francophones (JSF)*, Tokio, Japan, 2004.
78. A. T. Samiloglu, V. Gazi, A. Bugra Koku. Comparison of three orientation agreement strategies in self-propelled particle systems with turn angle restrictions in synchronous and asynchronous settings. *Asian Journal of Control*, 10(2):212–232, 2008.
79. S. Souissi, X. Défago, M. Yamashita. Using eventually consistent compasses to gather memory-less mobile robots with limited visibility. *ACM Transactions on Autonomous and Adaptive Systems*, 4(1):1–27, 2009.
80. O. Soysal, E. Bahçeci, E. Şahin. Aggregation in swarm robotic systems: Evolution and probabilistic control. *Turkish Journal Electrical Engineering*, 15(2):199–225, 2007.
81. K. Sugihara, I. Suzuki. Distributed algorithms for formation of geometric patterns with many mobile robots. *Journal of Robotics Systems*, 13:127–139, 1996.
82. S. Susca, S. Martinez, F. Bullo. Monitoring environmental boundaries with a robotic sensor network. *IEEE Transactions on Control Systems Technology*, 16(2):288–296, 2008.
83. I. Suzuki, M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.

84. T. Suzuki, R. Sugizaki, K. Kawabata, Y. Hada, Y. Tobes. Deployment and management of wireless sensor network using mobile robots for gathering environmental information. In: *Proceedings of 9th International Symposia on Distributed Autonomous Robotic Systems (DARS)*, pages 63–72, 2009.
85. O. Tanaka. Forming a circle by distributed anonymous mobile robots. Technical report, Department of Electrical Engineering, Hiroshima University, Japan, 1992.
86. O. Tekdas, J. H. Lim, A. Terzis, V. Isler. Using mobile robots to harvest data from sensor fields. *IEEE Wireless Communications*, 16(1):22–28, 2009.
87. G. Wang, G. Cao, P. Berman, T. La Porta. A bidding protocol for deploying mobile sensors. *IEEE Transactions on Mobile Computing*, 6(5):563–576, 2007.
88. G.Wang, G. Cao, T. La Porta. Movement-assisted sensor deployment. *IEEE Transactions on Mobile Computing*, 5(6):640–652, 2006.
89. P. K. C. Wang. Navigation Strategies for Multiple Autonomous Mobile Robots Moving in Formation. *Journal of Robotic Systems*, 8(2):177–195, 1991.
90. M. Yamashita, I. Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science* (to appear), 411(26–28): 2433–2453, 2010.
91. J. Yu, M. LaValle, D. Liberzon. Rendezvous without coordinates. In: *Proceedings of 47th IEEE Conference on Decision and Control*, pages 1803–1808, 2008.
92. Y. Zou, K. Chakrabarty. Sensor deployment and target localization in distributed sensor networks. *ACM Transactions on Embedded Computing Systems*, 3(1):61–91, 2004.

Part VII
Security Aspects

Chapter 22
Security and Trust in Sensor Networks

Przemysław Błaśkiewicz and Mirosław Kutyłowski

Abstract The concept of security for tiny artifacts has been studied in a wide range of aspects, from authentication through data integrity to intrusion detection. This chapter provides a broad overview of some of the techniques developed for constrained devices where computational power, memory capacity, and energy limitations enforce slightly different approaches to these problems, when compared to standard high-end devices. In the following, we present ideas that leverage unique properties of sensor networks (also wireless sensor networks) to provide consistent and secure systems for information gathering and sensing.

22.1 Security in (Wireless) Sensor Networks

When thinking of any modern computer system, in most scenarios some level of concern about its security comes into the spotlight. From a simple authorization of data through illegitimate access discovery and prevention to obfuscation and privacy of communication—security aspects are becoming inherent characteristics of modern digital world. In what follows, wireless sensor networks will be our world, with all their possible functionalities, constraints, and systematic features influencing our perception of security and trust. On the other hand, we will be faced with all problems that follow from lack of direct physical control over such networks, their diversity, and rapid evolution.

Model

We shall focus on a more-or-less adequate model of the network, for which solution ideas will be presented. We assume a network of nodes communicating via shared radio channels, with their own identities and some small storage and computation capabilities. As single entities, nodes are capable of gathering information and reporting it to one another or some high-end station. As a system, they are able

P. Błaśkiewicz (✉)
Wrocław University of Technology, Wrocław, Poland
e-mail: przemyslaw.blaskiewicz@pwr.wroc.pl

to perform some more intricate operations, such as detection, alarming. A more detailed description is given elsewhere in this book, and for our use the above model is enough as a starting point.

22.1.1 Types of Attacks

First of all, let us describe some feasible attacks that are of particular interest for sensor networks.

Node capture. In many scenarios nodes operate in an open environment without any supervision. The attacker can simply pick out a node and run any form of cryptanalysis on its material; this can leak all secret data stored in the node or allow reconfiguration so that it serves the attacker. Of course, malicious interception of *all* nodes from the network is also possible, but here we focus on an adversary that wants to utilize the network for his own purposes.

Sybil attacks. For Sybil attack, an adversary introduces multiple identities for one physical node. Even though he does not gain hardware advantage, it enables him to undermine the protocols, such as leader election, voting. Sybil attacks are in particular aimed against trust and reputation systems: if an adversarial node behaves in a wrong way and its reputation goes down, then it can leave the network and return with a different identity.

Cloning attacks. By cloning we mean inserting multiple physical nodes with the same identity. This provides hardware advantage (e.g., multiple locations). In a hybrid approach, the malicious nodes gain their own identities. Cloning attacks may be a consequence of node capture: in this case the original node may be still functioning, but an adversary deploys additional nodes with the same identity for his own purposes.

Wormhole. In this scenario the attacker utilizes an additional communication channel to capture the data in one point of a network and present it in another, and vice versa. Of course, this introduces confusion in all protocols relying on nodes' location, such as terrain monitoring or topography discovery.

Key interception. A key can be compromised either by means of physical extraction from a captured node, or by breach in protocol, including cryptanalysis. A particular challenge here arises from the fact that the attacker can be using powerful equipment which by far outperforms simple hardware to be found on sensor nodes. With intercepted keys the adversary may threaten communication, create fake nodes.

DoS. The Denial of Service attack renders a node, part, or whole network inoperable due to overloading it with unnecessary operations. Attacks of this kind often are performed on protocols where in normal operation some heavy computations are executed as occasional but indispensable element. An attacker controlling many nodes may perform a distributed DoS, which is by its nature more difficult to localize and lock out. Furthermore, DoS attacks can target radio communication links, blocking exchange of useful information.

Replay. Some protocols are stateless, i.e., one execution of an algorithm can be repeated at a later moment to yield proper results. This allows an attacker to record messages at one instance and replay them later on or at different locations. Note that even if authorization algorithm is in use, the attacker is in possession of authorized messages.

22.1.2 Threats

Of course, performing an attack on a network should have its purpose. In the case of wireless sensors this can be twofold: either disabling the network or accessing and/or distorting data the network reports. Depending on a case, either form may be chosen by an attacker, resulting in some of the following outcomes:

Battery drainage. In most cases, the sensors are battery-run devices. This limited source of energy can be prematurely used up for bogus operations and extra communication, induced by the attacker. A drained battery renders given node useless, requiring replacing the batteries or recharging them. Moreover, the workload from a disabled node, when shifted to other nodes, can cause an avalanche effect and prematurely disable other nodes.

Network control loss. A successful attack can lead to compromising a number of nodes in the network, if not the entire network at all. The compromised nodes can perform further attacks, thus spreading the attacker's control, or render the network less efficient or even useless. Taking back the control over the network, if possible at all, can entail such cumbersome activities as physical collection of compromised nodes and re-programming them back to the original state.

Data leakage and ambiguity. If an attacker can make the network leak some useful information or influence it to perform his own algorithm this poses a serious threat to the network's owner. On the other hand, an arbitrary or random data injected to the network can in fact render the network useless in that sometimes noisy or uncertain information is worse than no information; especially if the breach is not detected.

Goals

Typical goals for security and trust, which we will discuss in this chapter, are the following:

Authorization. This applies to the devices. In short—an authorized device is one that has right to operate within the network, obtain access to its resources, and communicate with other stations.

Data authentication. Data authenticity is the concept that follows device authorization. Namely, an authorized station is assumed to provide authentic (i.e., true) data. However, since wireless communication is open (at least in terms of medium access), everyone can potentially transmit information or disturb messages in the network. Authenticity paradigm assures that the received data either originated from authorized device or is disregarded.

Data secrecy. As mentioned above, the wireless transmission can be easily eavesdropped. Therefore, means must be taken so that important information is unintelligible for everyone except the recipient. *Forward secrecy* assumes that should the protection scheme be broken into, all future messages will continue to be safely transmitted. On the other hand, *backward secrecy* assures that all messages previous to the breaking of the scheme remain safe.

Compromise detection. Finally, schemes are designed to reveal illegitimate behavior in the network. Depending on the type of fraud, adequate actions should be taken.

In the following sections recently proposed or widely used mechanisms for securing some aspects of wireless sensor networks are presented. Detailed descriptions are omitted; instead, we put particular stress on inherent and emergent properties of sensor networks that make some approaches more feasible than other. It seems that despite new ideas that show up every year in the field, these general guidelines remain the same and lay at the basis of many protocols.

22.2 Information and Node Authentication

A continuous flow of information in the network often requires, depending on the application, that data be authenticated, i.e., its origin is known and legitimate. This can be achieved by means of shared key authentication, where the originator of the information uses a secret to mark the data in a way that allows its verification by all other owners of the secret. A serious drawback of this approach becomes clearly visible when more than two parties take part in the communication. This would either require each subset of communicating parties to share own secret, or that each two parties share a secret and the message is encrypted/decrypted on a hop-by-hop basis along its route from the source to the recipient. The former solution can, in the worst case, result in an exponential growth in storage requirements, while the latter rises the computation requirements for each sent message linearly with the number of parties relaying the message. On the opposite extreme, one may have one common secret among all. This however makes it then possible for any party to forge a false message without being recognized or simply losing the secret whenever it is leaked from one single device.

Another approach would be to implement solutions based on asymmetric cryptography. There, each message would be *signed* by the issuer in a way that makes it possible to *verify* its validity. The difference between this and the previous solution is that each entity has two *keys*: one, secret used for authentication and the other, made publically known, for verification by the information recipients. For such solutions, however, problems such as availability of public keys or dealing with compromised secret keys exist. Also, computations required in such *asymmetric* cryptosystems usually are more complex and include large numbers and/or complex algebraic structures arising feasibility problems in terms of storage and timing.

Below we present an overview of standard contemporary techniques for authenticating traffic in a network of, by design small and slow, sensor nodes. The sections below describe protocols allowing secure generation of *message authentication codes* (MACs), establishing common secrets and signature creation and verification. The selection is made to present possibly different approaches to the problem and introduce techniques from different corners of cryptology.

22.2.1 Chaining Protocols

In a node-to-node communication over a multi-hop path there is always a possibility of data being altered by intermediate nodes or even simply substituted with another information. In essence, this can be alleviated by using MAC (*Message Authentication Code*): a form of a fingerprint of the message that can be calculated by both communicating parties. An obvious condition here is that the MAC should not be computable by any of the other parties *at the time of transmission*, which would obviously let them change the message along with its associated MAC on the fly.

The main trick of chaining protocols is to connect data packages transmitted at different moments into a chain such that it becomes infeasible to remove or add a package into a chain without detection. Once a chain gets started, it authenticates itself so that we can trust the whole chain if we can trust any of its elements. To do this, the sender (originator) of the message appends some secret value K to a message M and only then does he compute $\mathrm{MAC}(M, K)$. On receipt, the receiver cannot verify the MAC without knowing the secret used to generate it, so it has to wait for it (or request it) from the sender. Note that the secret K can then be sent out in plain text as it is only to verify the validity of the message already transmitted.

This concept is often referred to as *deferred disclosure*, wherein the information used to calculate some validity codes is revealed after the codes have been computed and transmitted. Noticeably, deferred disclosure can be implemented with relatively simple operations available on weak devices.

22.2.1.1 μTesla

The assumption that data do not have to be authenticated in an on-line fashion at the moment of reception lies at the heart of TESLA [44] protocol by Perrig et al. In fact, the messages are sent with their MACs and temporarily accepted and stored in receiver's buffer. The MAC is generated using some value r (by applying *hash chain*, see below) at first known only to the sender, so it cannot be verified at the moment of reception of the message. However, the protocol assures that r is revealed on timely basis, thus allowing verification of all MACs. After a timeout without hearing r the messages marked using r are dropped as invalid.

The idea got further extended into modified TESLA [45], making the protocol more suited to wireless sensor network applications. The authors introduced techniques such as Merkle trees or hash chains to enable time synchronization or DoS attack prevention (see [45] and the rest of this chapter). However, perhaps the most instructive solution is that dealing with packet buffering requirement.

Primarily, TESLA is a protocol for broadcast authentication; as such, in the world of sensor networks the sender can be associated with a more powerful base node. Assume that a node sends t packets with data chunks authenticated with the same r. The hash value of the *next* packet payload is appended to the *current* one, and the MAC is calculated over their concatenation. Namely, the jth packet P_j containing the jth data chunk M_j and authenticated by the ith instance of $r = K_i$ is the following tuple:

$$P_j = \langle D_j, \mathrm{MAC}(K_i, D_j), K_{i-1} \rangle, \quad \text{where } D_j = (M_j \| H(M_{j+t}))$$

After t packets have been sent, the current authorizing key r changes to be $r = K_{i+1}$ and the packets have the form:

$$P_{j+t} = \langle D_{j+t}, MAC(K_{i+1}, D_{j+t}), K_i \rangle$$

Observe that packet P_{j+t} contains K_i: the key used to authenticate previous t messages, which enables the receiver to perform validity check. If this succeeds, it follows that $H(M_{j+t})$ is also authentic, and so is the whole P_{j+t}. As we see, there is a *two-way dependency between packets* and one can really say that they are *chained*. This dependency between packets can be extended by adding extra fields with adequate hashes in each packet. With this assumption, the protocol provides a way of on-line message authentication without requiring the nodes to buffer too much information. This responsibility is shifted to the sender.

Further modifications to TESLA resulted in μTesla [46]. As before, the messages are MAC'ed by consecutive hash chain keys K_i on time-epoch basis, i.e., the ith key is used to authenticate *all* messages in time interval i, but epoch duration is constant. Addressing the ad hoc nature of sensor networks, and keeping in mind limited memory of sensors, μTesla allows bootstrapping of a new device at any time. To join the network, a node needs to verify the authenticity of *one* (any already used) key from sender's key chain and get loosely time synchronized. To this end, the new device sends a join request containing a nonce N_M. The sender responds to the request with the following nonce:

$$\langle T_{\mathrm{s}} \| K_i \| T_i \| T_{\mathrm{int}} \| \delta, \mathrm{MAC}(K_{\mathrm{MS}}, N_M \| T_{\mathrm{s}} \| K_i \| T_{\mathrm{int}} \| \delta)\rangle$$

where T_{s} is current sender's time, K_i is a key that was used starting at time T_i for a period of T_{int} (epoch duration); δ is a parameter describing time interval to elapse after an epoch end before this epoch's key is revealed. K_{MS} is a symmetric key shared between the sender and the new device; the protocol allows only for a limited number of sender–receiver pairs in the network, thus saving memory for storing such symmetric keys. The joining node can, using the information from sender, establish the beginning of the next (say, jth) epoch and start receiving. When K_j is revealed, the receiver should be able to rebuild $j - i$ steps of the hash chain from K_i to K_j. If it cannot, it should drop messages from epoch j and restart the join procedure.

22.2.1.2 Hash-Chained Authentication

The idea of a *hash chain* is closely related to TESLA protocol, as it provides a good mechanism for generation of secrets for deferred disclosure. In the naïve approach this value could be taken to be any random string. This solution however suffers from a serious drawback: there is no way of telling who in fact sent the message. It follows that the secret value must be uniquely associable with a given entity in the network. A one time solution would be to use a secret (key) shared by the two communicating parties, but after a single authentication the secret should be renewed, due to replay attack possibility. Updating the key brings up the problems of key establishment (cf. Sect. 22.3.6.1). Moreover, in a many-to-one type of communication the receiving party would have to store the secrets for each possible sender—a solution that is unacceptable in many lightweight applications.

The basic scheme of ALPHA protocol by Heer et al. [23] assumes the use of hash chains designed as follows. First, the sender (S) chooses a random value h_0, and S computes $h_1, h_2, \ldots, h_n$ using the equation

$$h_i = H(h_{i-1})$$

where $H(\cdot)$ is a one-way hash function (which means in particular that it is infeasible to compute preimages for H). Parameter n depends on the storage capabilities and foreseen demand for authentication operations. Each result is stored in memory and the last element h_n (the *anchor*) is made known (in a secure way) to the receiver

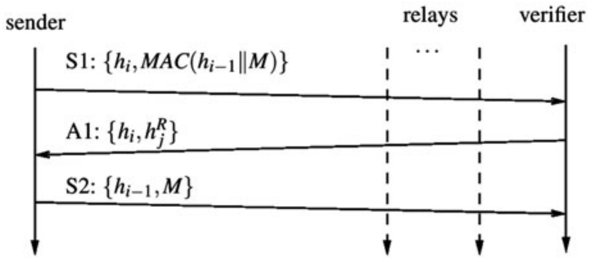


Fig. 22.1 A two-way authenticated message passing scheme using ALPHA. Relaying nodes can verify the values h_k in transmissions S1 and A1 and refuse to route the corresponding packet in case the values do not match the hash chain of the corresponding transmitting node

(R). Note that if h_i is known to the receiver, it can verify the sender when presented with h_{i-1} by checking

$$H(h_{i-1}) \overset{?}{=} h_i$$

as a hash cannot be inverted by design.

Sending of an $(n - i)$th message M involves the following operations (see also Fig. 22.1):

1. sender S generates MAC$(h_{i-1}\|M)$ and sends it along with h_i to receiver R (transmission S1);
2. R acknowledges replying with h_i and its own hash chain element h_j^R (transmission A1);
3. S discloses the next element of his hash chain h_{i-1} and sends it along with the message M (transmission S2).

Of course, R has committed to its hash chain anchor beforehand and reveals its consecutive elements to authenticate its acknowledgments. A two-way authenticated channel is constructed. However, intercepting the data in message S2 (specifically, the value h_{i-1}) allows an attacker to authenticate the next message S1 as if from a legitimate sender. Even though injecting an arbitrary message M is not possible, the attacker can force the transmitting nodes into storing and running calculations on bogus messages.

To counteract this, the scheme is extended by using two distinguishable hash chains: one to provide authentication in Step 1 and the other for generating MAC secrets. Moreover, to enable two-way reliable transmission, the responses of R can include precomputed acknowledgments and non-acknowledgments, later on disclosed in the fourth, additional step of the algorithm:

1. S generates MAC$(h_{i-1}\|M)$ and sends it along with h_i to R.
2. R acknowledges by replying with h_j^R and provides two hashes: $H\left(h_{j-1}^R\|1\|s_{\text{ack}}\right)$ and $H\left(h_{j-1}^R\|0\|s_{\text{nack}}\right)$, for acknowledgement and refusal, respectively; the random strings $s_{\text{ack}}, s_{\text{nack}}$ are to obfuscate the result.

3. S discloses the next element of his hash chain h_{i-1} and sends it along with the message M.
4. R verifies M against received MAC and responds disclosing its next authentication chain element h^R_{j-1} and either s_{ack} or s_{nack} required to compute $H(\cdot)$ from Step 2 above.

By using the chain it is possible to issue *pre-signatures* for large amounts of data. If such data is to be sent, it first needs to be split into a number of packets, but it is more convenient if all the parts are sent in one burst transmission. Therefore, all signatures for the packets are generated with the same undisclosed element of a hash chain and sent en masse in Step 1 of the algorithm. Next, without ACK message from receiver, the data packets can be sent out at high rates. Now, since the receiver already has the signatures for the pieces of data it is receiving, it can verify their validity on the fly and issue an ACK only after the burst transmission is finished. On the other hand, should any of the parts be discovered invalid, the receiver can instantly reject the entire transmission.

Mechanisms of ALPHA allow also for en route authentication and bandwidth adaptation. Specifically, each relaying node can maintain current state of hash chain for all nodes routing traffic through this relay. After receiving such packet, the relay can check if the authenticating part (h_i in Fig. 22.1) of each transmission is indeed a part of the hash chain of the node originating the message. A natural profit for the network is that unwanted (injected) traffic is simply dropped at the very first relaying node, saving the bandwith for legitimate routing. In other words, at the expense of some additional overhead for storing current state of hash chains of transmitting nodes, the network can be protected against flooding with messages. This is a good illustration of emergent inter-network cooperation to be observed in wireless sensor networks.

22.2.1.3 Merkle Trees Authentication

As with the last example above, authenticating multiple messages with hash chain values can introduce additional overhead for storing the pre-signatures of each data item. Note that if data are split into n parts, then the receiver needs to be able to store all n pre-signatures that is sent in Step 1 of the algorithm. While this approach requires memory size linear to the number of signatures (n), the Merkle tree (MT) can limit this requirement to $\log(n)$. MT is a plain binary tree, where the leaves store hashes of chunks of the information to be authenticated and each node has a label value calculated to be the hash of the concatenated values of its two children. That way the root of the tree depends on each leaf, but in order to verify the validity of a single leaf only $\log(n)$ values are needed, n being the number of leaves in the tree. These values are siblings of the nodes on the path from the leaf to the root, and the root itself (see Fig. 22.2).

This mechanism is used in ALPHA ([23]) to alleviate the storage problem of multiple pre-signatures in high-rate transfers. There, a message is split in chunks and these are used as hash arguments to calculate values in the leaves. Additionally, the transmission is authenticated by means of a hash chain (see the paragraph

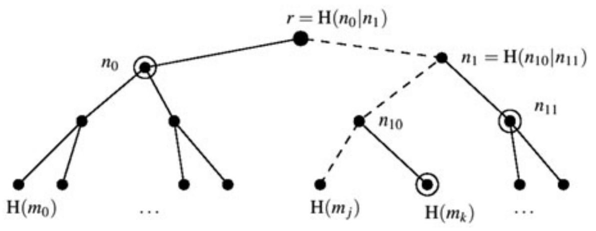


Fig. 22.2 Merkle tree for message $\mathcal{M} = m_0|m_1|\ldots|m_n$. To authenticate message chunk m_j only values of siblings (in *circles*) of nodes on the path (*dotted line*) to the root are needed

above). The sender calculates the values for nodes MT as described above, with the exception of the value of the root:

$$r = H(h_{i-1}\|n_0\|n_1)$$

where n_0 and n_1 are values of children of the root, and the h_{i-1} is the next-to-be-revealed value of the sender's hash chain. So calculated r is sent as the pre-signature of the transmission in Step 1 of the algorithm (see the previous section). Next, each chunk of the message is sent along with h_{i-1} and all information needed to authenticate it by means of the MT. Here, the overhead in storage is balanced with overhead in transfer. Moreover, the messages can be sent out in a carefully designed sequence in order to allow efficient caching of the values of the nodes in MT.

Slightly different use of Merkle trees is described in [40]. There, Miller and Vaidya present a system where a MT is generated globally by a trusted third party. The leaves of MT contain unique identifiers of nodes combined with a Bloom filter[1] with hashed in all keys a given node is preloaded with. In such scenario, the MT provides a lightweight and efficient means for verifying the validity of a node–key pair. Each node is loaded with root value of the tree and the set of nodes necessary to authenticate its own leaf value. In the course of the protocol the Bloom filter is broadcast along with credentials from the MT in order to allow other nodes to verify the validity of the Bloom filter and hence the validity of the identity of the node. (For an in-depth description of Miller and Vaidya protocol refer to Sect. 22.3.2.2.)

22.2.1.4 Maintaining Link Consistency

A selection of lightweight mechanisms for maintaining secure channel over a number of consecutive sessions is presented in [8]. In general, the assumption is to generate some information in an impossible to guess way and exchange it during

[1] A Bloom filter [7] is a bit-vector of length n, originally set to all-zeroes. To insert a value into the filter one calculates a hash function $H : \{0, 1\}^* \rightarrow \{1, \ldots n\}$ and sets the corresponding bit in the filter to one. At later time it is possible to effectively check if a given value *does not* belong to the filter by calculating the $H(\cdot)$ on the value; on the other hand, due to possible collisions in the hash function, the Bloom filter is liable to *false positives*.

initialization phase. Then, in each transmission, new authorization keys are derived from this secret which guarantees logical continuity and consistency of the link.

The first technique employs a chain of keys, where the key for the next transmission K_{next} is sent under current key K_{now} encryption. To countermeasure the threat of leaking a single key that would lead to compromising all next transmissions, two secret seeds, $s1$ and $s2$, are exchanged during initialization phase. Then, both keys are additionally masked with these seeds and the final ciphertext transmitted takes the form:

$$\text{Enc}_{K_{\text{now}} \oplus s1}(K_{\text{next}} \oplus s2)$$

where $\text{Enc}_A(B)$ denotes the ciphertext of B obtained with key A. Compromising encryption keys $K_{\text{now}} \oplus s1$ from consecutive transmissions leads to compromising all future encryptors; yet the keys K_{now} remain secure as long as seed values remain secret.

To the same avail, one can use one-way functions. In the initialization phase a product N of k large primes is exchanged between parties. To logically link consecutive transmissions, particular factors of N are revealed. Since the problem of integer factorization is computationally hard, guessing correct value is difficult but simply checking if the revealed number is a divisor of N allows a quick authentication of the sender. Also, the receiver may request additional random factors to prevent lucky-guesses and replay attacks.

Authorizing n sessions using the concept of *logarithmic keying* requires generation of $2 \log n$ symmetric keys, grouped in pairs $\left(K_j^0, K_j^1\right)$, $j \in (0 \ldots \log n)$. The keys, along with a constant message M, are committed to during initialization. The kth session is identified by the binary representation (i_j), $j \in (0, \log n)$ of k, and the authentication key for session is generated by XORing all keys $K_j^{i_j}$. The key is used to hash M and the resulting value is sent as authenticator for the current session. Note that compromising the hash function does not reveal any secret keys so less requirements can be placed on this potentially computation-demanding operation. Also, the parameter n depends exponentially on the number of keys that need to be stored, a feature that allows for good scaling of the scheme.

22.2.2 Asymmetric Methods

22.2.2.1 Rabin Scheme

When data is sent from computationally constrained tiny nodes to a more powerful device such as a base station one can utilize this disproportion and apply similarly asymmetric operations to assure secrecy or security of data. In fact, one can think of a normal operation of a sensing network that gathers data and forwards it in a secure (encrypted) manner to a base station and receives authenticated commands in return. Results published in [21] show that Rabin scheme with carefully picked parameters and slight improvements can result in a very effective method for encryption and signature verification. At the same time, decryption and signature generation times are prohibitively long (the difference is almost by factor 500). Similar conclusion is to be found in [49], a recent survey of cryptographic primitives for sensor nodes.

In Rabin scheme the private key is a pair of large distinct primes (p, q), and the public key is their product $n = pq$. The ciphertext of message M is then

$$c = M^2 \bmod n$$

An inherent property of this operation is that the same value c can be generated for four different messages M and additional information is needed to select the correct one. Decryption is by far more complex and requires knowledge of both p and q, because general solution for M is $M \equiv \sqrt{c} \bmod pq$, and this can only be solved if the decomposition of the modulus is known. Essentially, the problem boils down to calculating *square roots* (mod p) and (mod q) and, by Chinese reminder theorem, reconstruction of the four possible messages. This calculation can be particularly simplified by setting $p \equiv q \equiv 3 \bmod 4$ (cf. [21]), but it still remains challenging for a small sensor node.

Similar problems arise in signature schemes. There, the public key is a pair (n, b), $b \in \{1, n\}$, secret key is (p, q) as before. Additionally, the scheme utilizes a hash function $H : \{0, 1\}^* \to \{0, 1\}^k$. To sign a message M one appends a random string u to it and attempts to solve

$$x(x + b) = H(M|u) \bmod n$$

which is an instance of the decryption problem described above. Only for one in four values of u can the solution be found, and this process runs with different u until x is calculated. The signature is then the pair (u, x). Verification is as simple as comparing if $x(x + b)$ and $H(M|u)$, computed with values obtained from the signature, are the same.

The scheme requires squaring to generate a ciphertext and hash function and multiplication to verify a signature. Both these tasks can be easily performed in low-end nodes, while the more demanding operations are shifted to more powerful devices.

22.2.2.2 Diffie–Hellman Key Exchange

Here we briefly describe how the Diffie–Hellman protocol allows two nodes communicating in open text to agree on a common secret.

The protocol assumes that two parameters are publically known: the generator g and a prime modulus p. The run of the protocol is as follows:

1. node A generates a random value r_1 and blinds it by calculating $x_A = g^{r_A} \bmod p$; this value is sent to node B over an insecure channel;

2. node B performs identical operation with its own random value r_B to obtain x_B and transmits it to A;
3. both nodes calculate common secret to be

$$(x_A)^{r_B} \bmod p = (x_B)^{r_A} \bmod p = g^{r_A * r_B} \bmod p$$

For the scheme to be sound we need to assume that *discrete logarithm problem*, DLP (i.e., extracting x from g^x), is computationally hard. Then in particular extracting r_A from x_A or r_B from x_B is infeasible. Therefore, an eavesdropper cannot calculate the common key as she is only in possession of part of the needed information. Hardness of DLP is a frequently used assumption in cryptographic schemes.

The tinyPK [52], a set of public key-based protocols developed in TinyOS for Mica2, utilizes DH protocol for establishing common secrets between two motes (see Sect. 22.2.2.6). Setting $g = 2$ resulted in 80 s operation time for 1024 bits long modulus and 30 bytes exponent size for exponentiation in Steps 1 and 2 and 110 s operation time for the exponentiation in Step 3. As this may seem prohibitively long, the parameters here provide strong security level as considered in standard, unconstrained machines. At the same time, taking the modulus to be 512 bits long and exponent 16 bytes keeps both timings well below 20 s, which may be more feasible for securing traffic in wireless networks. Moreover, one can use Diffie–Hellman only for initializing a connection between two nodes, later they can use a shared secret to derive a session key with fast methods.

22.2.2.3 Polynomial-Based Authentication

A slightly different method, also employing exponentiation but to smaller exponents, involves use of polynomials [55]. The polynomials here are secrets, and message verification is performed by comparing the offered value of the secret polynomial at given points to the value calculated based on the message fed into a verification polynomial. Specifically, the secret bivariate polynomial coefficients $A_{i,j}$ are chosen randomly from finite field F_q, where q is prime and d_x and d_y are degrees of x and y:

$$f(x, y) = \sum_{i,j} A_{i,j} x^i y^j, \;\; 0 \le i \le d_x, \;\; 0 \le j \le d_y$$

$f(x, y)$ is stored at the base station along with a hash function $H(\cdot)$. Each sensor stores its ID = u along with a *verification polynomial* ver $f_u(y) = f(u, y)$ and the same function $H(\cdot)$.

Sending an authenticated message M follows these steps:

1. $f(x, y)$ is evaluated at $y = H(M)$ to yield an univariate polynomial of x, called *message authentication function*:

$$\mathrm{MAF}_M(x) = f(x, H(M))$$

2. a tuple $\langle M, \mathrm{MAF}_M(x)\rangle$ is sent;
3. on receipt, a sensor calculates its verification value for $y = H(M)$:

$$\mathrm{ver}\, f_u(H(M)) = f(u, H(M))$$

4. the received $\mathrm{MAF}_M(x)$ is evaluated for $x = u$ and the result compared to the one obtained in Step 3. Message authenticity is confirmed by their equality.

An obvious problem here is that coefficients of $f(x, y)$ can be interpolated either by obtaining more than d_y MAFs for different messages, or by combining the knowledge of more than d_x nodes. One way to solve this would be to increase both d_x and d_y parameters, which would lead to significant overhead in storage (coefficients of ver $f_u(y)$) and transmission (coefficients of MAF(x)). The authors employ probabilistic solution introducing random perturbations to calculations done on communicating devices. Specifically, to prevent interpolation of $f(x, y)$ from MAFs, the latter ones get perturbed by a small factor picked at random from F_q. Also, the ver $f_u(y)$ for each node is not given exactly but perturbed by a small number from F_q. The verification at the node then changes to checking if the *difference* between computed verification code and the one sent along with the message is in a given set, dependent on the parameters of the distortion, given as system parameters.

So far, the proposed schemes leveraged bivariate polynomials and allowed for one-way communication of a trusted station to other nodes. This idea can be further extended to accommodate bi-directional authenticated communication by means of tri-variate polynomials. However, this extension involves more intricate operations since naïve approach leads to slight security flaws. For specifics of this process see [55], while here we state that the scheme for bi-directional communication uses two different polynomials (perturbed with carefully chosen polynomials, instead of numbers as in the second scheme) for verification and authentication.

The last scheme was implemented on Mica2 motes. Timings obtained for authentication and verification are much smaller than those offered by, e.g., tinyPK library [52] or elliptic curve cryptography and only slightly worse than those obtained for Rabin scheme and NtruEncrypt [21]. The two last schemes however produce larger signatures which in turn increase communication overhead.

As a remark, it is worth noting that [55] proposes a novel way of dealing with problems of limited capabilities of sensors by employing probabilistic techniques. This merging of probability with basic algebra allows for relaxation of constraints (i.e., minimal degrees of polynomials that provide measurable security) while keeping the properties of the system within frames of usability. Another idea for exploiting polynomial shares is for pairwise key establishment [36]. We describe this algorithm in greater detail in Sect. 22.3.3.

Another direction is to perform all operations in the exponent, i.e., instead of $f(x, y)$ we use $g^{f(x,y)}$, instead of addition we use multiplication, and instead of multiplication by a known factor we use exponentiation. Computationally, this is much harder, but offers higher level of security, due to the fact that getting g^{ab} from g^a and g^b is infeasible. This technique has been used by many authors for constructing broadcast revocation schemes (i.e., broadcast schemes where some participants can be excluded so that they cannot decrypt the message received).

22.2.2.4 NtruSign

In a number of contemporary cryptographic systems the amount and type of executed computations make them prohibitively too complex for use in constrained devices. However, some systems build up on simple operations and use some level of probability to achieve comparable results in terms of security, but by far better performance in terms of execution times and, therefore, energy consumption. One such system is NTRU [25], where basic operations involve multiplication, convolution (that is, in fact, multiplication), and reduction modulo a reasonably small number. The system is built on a number of parameters: numbers N, p, q, where N is prime and p and q are coprime and $q \gg p$, and four sets of polynomials of degree $N - 1$: $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_\phi$, and $\mathcal{L}_m$. Its safety relies on difficulty of factorization of a polynomial into two polynomials with small coefficients, a problem relating to closest vector problem (CVP) and shortest vector problem (SVP) known for lattices [1, 39].

In *key generation* phase two polynomials, f and g, are selected randomly from $\mathcal{L}_g$, so that f has inversion modulo q and p. Let q^{-1} and p^{-1} be these inverses, respectively. The public key h is calculated as

$$h \equiv q^{-1} \circledast g \pmod q$$

where $\circledast$ represents cyclic convolution of polynomials, i.e., the kth coefficient of $F \circledast G$ to be

$$(F \circledast G)_k = \sum_{i+j \equiv k (\mathrm{mod} N)} F_i \cdot G_i$$

The private key is set to be f. To *encrypt* message $M \in \mathcal{L}_m$ one chooses a polynomial $\phi \in \mathcal{L}_\phi$ and computes

$$e \equiv p\phi \circledast h + M \pmod q$$

This ciphertext can then be *decrypted* in two steps. First, a polynomial a is selected to have its coefficients in the range $(-q/2, q/2)$ and to satisfy

$$a \equiv f \circledast e \pmod q$$

Then, M can be decrypted by simply calculating:[2]

$$M = p^{-1} \circledast a \,(\bmod\, p)$$

The probabilistic criterion is that this reduction can lead to different results for certain combinations of a and p^{-1}. The authors argue that this can happen with small probability and provide methods to alleviate this problem.

Based on this scheme, a signing protocol NtruSign [24] was also proposed. In comparison to different public key systems with application to sensor networks presented in [21] it can be noted that protocols based on NTRU are relatively fast and liable to parallelization. In fact, most of the operations performed in the scheme are cyclic convolutions that can be effectively run in parallel for each coefficient. This drastically reduces execution time (by a factor up to 100), but has its reflection in power consumption. On the other hand, Roman et al. [49] report that NtruEncrypt can be implemented with as little as 3000 gates, which makes it particularly suited for low-end devices.

22.2.2.5 Elliptic Curve Cryptosystems

In recent years, Elliptic Curve Cryptography (ECC) has gained a lot of attention in application to security problems of wireless sensor networks. Schemes utilizing ECC employ smaller key and signature sizes and simpler operations as compared to, say, RSA, while providing the same desired features of Public Key Cryptography. In ECC [29, 41] it is typical to consider the cubic equation:

$$y^2 = x^3 + ax + b$$

where $a, b \in F_p$ are constant parameters and p is a large prime. Then, an elliptic curve over F_p is a set of all points (x, y) such that $x, y \in F_p$ satisfy the above equation, and a point in infinity, $\mathcal{O}$. Cryptographic systems using elliptic curves base on the *elliptic curve discrete logarithm problem*—ECDLP. This is an analogue to discrete logarithm problem: here the cyclic group is the set $\{\mathcal{O}, G, 2G, 3G, \ldots\}$ for any point G on the curve, and calculating k given G and kG is the ECDLP. The hardness of ECDLP allows setting a random w to be a private key and corresponding wG to be the public key, once parameters a, b, p and point G on the curve are agreed upon.

With these primitives as building blocks, ECC cryptosystem consists of a number of schemes for signing, verification, encryption, and key establishment. TinyECC [35], a library of ECC modules for sensors running TinyOS implements elliptic curve variant for Diffie–Hellman key establishment for DSA (digital signature algo-

[2] In fact this is a two-step process: first, a polynomial $a \equiv f \circledast e (\bmod\, q)$ is calculated such that its coefficients are in the range $< -q/2, q/2 >$, which cancels out $(\bmod\, q)$ operation. Then, the blinding polynomial $p \cdot \phi$ is reduced $(\bmod\, p)$ and by inverse p^{-1} the message $(\bmod\, p)$ is recovered. For detailed discussion, see [25].

rithm) and IES (integrated encryption scheme). A number of optimization tricks were implemented in the library and a user has free choice as to whether enable them or not. This is particularly important as the nature of such optimizations is that they trade one overhead with another, e.g., providing faster calculations at the extra cost of RAM or ROM memory.

Code Optimization

Reportedly, there have been many attempts to make the ECC schemes more feasible for sensor networks. One example is work by Blass and Zitterbart ([5]). The authors make a thorough analysis of features that different parameters of the scheme provide and contrast them with a particular hardware (an ATmega128 in their case, the microcontroller to be found in Mica2 motes). In the course of analysis, they pick key size to be 113 bits, perform some possible precomputations, and store results in ROM memory in order to save the main RAM storage, fiddle with the code organization and assembler code optimization switches. As a result, timings of 34 s for signing (ECDSA) and 68 s for verification were obtained.

Special Purpose Hardware

Even further acceleration can be obtained by designing special purpose hardware and delegating some parts of the computation to a satellite chip, just the same way as data aggregation is delegated to sensors mounted on a sensor mote. In [21], not only special values for p and m were chosen to facilitate modular scaling, but also all standard arithmetic operations were hardwired in serialized fashion. As a result, the signing time with ECDSA is reported to be 410 ms and verification—820 milliseconds. A specially tailored processor for ECC operations (multiplication and summation) is presented in [3], where particular focus was put on the area required for an implementation. A similar presentation was given by Wolkerstorfer ([53]).

In conclusion, it seems that purely software-based solutions to ECC remain infeasible for wide application. On the other hand, offloading the burden of specific calculations to a designated hardware and careful selection of parameters may lead to more realistic solutions. This trend is readily observable in the realm of cryptographic smart cards, where heavy computation is done in designated hardware, whenever such heavy cryptography is required.

22.2.2.6 RSA

RSA is one of the most widely used Public Key Cryptography schemes used for encryption and authentication. However, due to its specific calculations and relatively large size of operands, it has long been considered inadequate for tiny constrained devices.

Specifically, to derive keys, two prime numbers (the RSA numbers), p and q, are selected, and their product $n = pq$ is calculated. Next, the *public exponent* e is picked to satisfy

$$gcd(e, \varphi) = 1$$

where $\varphi = lcm(p - 1, q - 1)$ is the least common multiply of $p - 1$ and $q - 1$.

Similarly, the *private exponent d* is chosen to satisfy

$$de \equiv 1 \mathrm{mod} \varphi$$

The public key then is the pair (e, n) while the secret key is the triple (d, p, q). So far, all the information can be performed off-line and only the credentials be loaded onto devices. The two operations, encryption and decryption, is done on-line: encryption of a message M into ciphertext C:

$$C = M^e \mathrm{mod}\, n$$

decryption with secret value d:

$$M = C^d \mathrm{mod}\, n$$

As said before, the two operations and keys are complementary, which means that in order to calculate a signature S of a message M, it is enough to set:

$$S = \mathrm{hash}(M)^d \mathrm{mod}\, n$$

To verify, one applies public key and the first equation to extract the value C to be checked against the hash of the signed message.

It is suggested that the length of public key (n) be at least 1024 bits, and choosing e too small (e.g., less than 16 bits) can significantly weaken the scheme. Therefore the RSA arithmetics is so problematic in application to resource constrained nodes. In [27] Hu et al. present secFleck, a trusted platform module designed to cooperate with Flec sensor nodes. Basic functionality is implemented and software interface provided to perform operations such as symmetric and asymmetric de/encryption, signature calculation, and verification. Performance analysis shows that offloading execution to external hardware accelerates the RSA encryption by the factor of 10,000 and takes 55 ms for 2048 bit key and 16-bit exponent.

By comparison, TinyPK is a library of optimized code implementing RSA cryptography for sensor motes, limited to operations with the public key, as these require less computation [52]. The infrastructure of TinyPK includes a publicly trusted certification authority (CA) where all entities willing to get access to the network should obtain their credentials. This is done by means of signing (in fact, by encrypting) by the CA the public key of the entity. The CA's public key is deployed in each mote prior to deployment.

The protocol is run between a third party (TP) joining the network and a sensor node (SN) in the following steps:

1. TP generates a tuple consisting of a nonce (timestamp) n and a checksum of its public key; TP encrypts the tuple with its private key to form ciphertext $C1$.
2. TP sends $C1$ and its public key signed by CA to SN.
3. SN can decrypt TP's public key using CA's public key and, consequently, extract n and checksum of TP's public key from $C1$.
4. SN encodes a session key s_K and n with TP's public key and sends it to TP.
5. TP decrypts the message and validates n; both parties now share the same secret s_K.

The value n in the protocol is used to prevent replay attacks. Note that the above scheme ensures that TP gains access to the network only if it obtained adequate credentials from CA. However, the authorization in the opposite way is not achieved: the TP cannot know if the sensor is a legitimate network device. A standard approach to this problem would be to have the sensor issue a signature, which requires calculations on its secret key. These in turn are computationally challenging to be implemented in standard power-constrained nodes to provide effective solutions. Instead, the authors propose that sensor be preloaded with the following additional information:

Diffie–Hellman: public value $p_{DH} = g^r$, for some publically known g and a secret random integer r, precomputed by the CA prior to deployment of the node, along with its signature $\{p_{DH}\}^{CA}$ by the CA;

text identification: string ID_{text}, consisting some proprietary information such as manufacturer, serial ID.

Next, after s_K is established, Steps 1 and 2 of the above protocol are run again, and additionally TP sends the first public value for Diffie–Hellman protocol (see Sect. 22.2.2.2). After that the SN

- calculates the secret key s_{DH} according to Diffie–Hellman protocol, which proves its legitimate connection with CA;
- calculates message authentication function MAC under ID_{text} and the nonce n obtained from TP, using s_{DH};
- responds with tuple: $(p_{DH}, \{p_{DH}\}^{CA}, \text{MAC}(ID_{\text{text}}, n))$.

After these steps, TP can verify if s_{DH} was calculated properly with the public value signed by CA and extract node information from ID_{text}. This way heavy computations with private key are replaced by two tricks: (1) most information required for authentication is pre-computed and stored on the node and (2) credentials are in the form that is unique for each node, yet easily obtainable.

By comparison with fully fledged public key cryptography schemes, the TinyPK lacks the ability to revoke credentials in case of private key compromise. This is normally achieved by means of *certificates* and certificate lists maintained by designated servers. This approach is however not feasible to implement in sensor networks both in terms of infrastructure and computation overhead.

22.2.3 Sensing Mobile Artifacts

In some cases the sensor nodes are not only sensing the environment but also other artifacts of the network. A good example of such an architecture are networks sensing mobile objects equipped with electronic tags: a batteryless electronic devices capable of storing small amounts of data and performing basic operations. Their performance is extremely limited, as they gain all the energy needed for operation via radio inductance only when confronted with a reader. The tag readers in turn are powerful nodes, with an efficient communication and information infrastructure, often carefully designed and maintained.

Depending on the application, such a network might demand strong privacy guarantees, reliable authentication, prevention against cloning, and so on. At the same time the mobile nodes should be as simple as possible. There are many efforts to provide appropriate algorithms and protocols of this kind (e.g., recall NtruEncrypt [21] mentioned earlier in Sect. 22.2.2.4). On the other hand, the system design is facilitated by the fact that the mobile devices communicate with a powerful infrastructure and not between themselves.

In this section we discuss two extreme cases of this kind: lightweight nodes that cannot be equipped even with symmetric cryptography, and powerful chips such as embedded in electronic travel documents.

22.2.3.1 Lightweight Node Protocols

The main problem we are focusing on in this section is designing a protocol that would be immune against an adversary that may eavesdrop the whole communication but does not capture the devices. The adversary's goal is to clone a device based on information gained from the radio channel. Of course, if a tag is merely transmitting its ID (as in case of simple RFID tags), breaking the system is immediate.

HB Family of Protocols

The background of the HB (Hopper-Blum) protocols is hardness of the problem of *Learning Parity with Noise* (LNP). Namely, let x be a secret bit-vector, A be a random binary matrix, and v be a noise vector where each single bit is set independently to 1 with probability ε. The goal is to find x given A and y, where $y = A \cdot x \oplus v$ and $\oplus$ is a bitwise XOR operation. For $\varepsilon = 0$, finding x is a trivial linear algebra task. On the other hand, for $\varepsilon = 0.5$ the vector y is fully random, and therefore it brings no information on x. For authentication purposes we use intermediate values ε such that a sufficiently large fraction of bits in y coincide with $A \cdot x$, but LNP is still hard.

The HB authentication protocol [26] assumes that a tag and a reader share a secret x. The reader performs the protocol to check that it is talking with the tag holding x. The protocol consists of N trials. Within a single trial:

1. the reader sends a challenge a;
2. the tag responds with $z = \langle a, x \rangle \oplus v$, where $\langle a, x \rangle$ is the scalar product of a and x, $\oplus$ is the XOR operation, and v is a noise bit equal to 1 with probability ε;
3. the reader counts the trial as successful if the response z equals $\langle a, x \rangle$.

Finally, the reader checks if the number of successful trials does not deviate too much (according to the rules of statistics) from $N \cdot (1 - \varepsilon)$.

HB protocol can be used for identification of tags. In this case the reader performs computations for each secret x stored in its database, which can become a serious problem in large-scale systems. Another problem is that a malicious reader can easily retrieve secret x from a tag. Namely, in order to remove the noise it sends the same challenge a many times. Majority of the answers show the value $\langle a, x \rangle$. This problem is addressed by HB+ protocol, where the tag and the reader share two secrets, say x and y. During a trial, the tag determines the second random challenge b and the answer of the tag equals $\langle a, x \rangle \oplus \langle b, y \rangle \oplus v$. Still, this does not solve all problems – man-in-the-middle attack is still possible. Regarding algorithm efficiency, one can perform all trials in parallel avoiding the overhead due to interaction. However, the communication volume is still high.

Another variant of HB protocol is HB# [22], where the secrets are the matrices X and Y, and the answer of the tag equals

$$X \cdot a \oplus Y \cdot b \oplus v$$

for challenges a and b. This reduces communication volume, since the challenges are two single vectors and not a set of vectors.

Hidden Subsets

Another idea to provide untraceable identification with simple means is to use *hidden subset identifiers* [14]. The idea is that the ID returned by a tag is a random string generated by tag's hardware, except for a few bits that are not random. Specifically, they are computed as XOR of a (hidden) subset of bits from other secret positions (see Fig. 22.3).

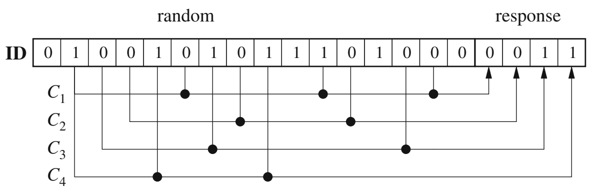


Fig. 22.3 Hardware setting for four-bit-dependent part calculation with hidden subsets C_1, C_2, C_3, and C_4. XOR operators are represented by *black circles*, random part is filled with reader's challenge. Note that for example, the first bit of **ID** is independent

Strictly speaking, the bit positions in an ID string are independent and dependent ones. The location of the dependent positions is secret. For the ith dependent bit b_i we have a subset C_i of independent positions; C_i is secret, too. Once a tag becomes activated by the reader, then

- the bits in independent positions are set at random by hardware,
- for each dependent position, with probability p the bit b_i is set to the XOR of all bits in C_i, otherwise it is set at random.

Probability p is substantially higher than $\frac{1}{2}$, but also substantially lower than 1. The reader knowing all dependent positions and the hidden subsets computes for how many i's b_i equals XOR of the bits in C_i. If the fraction of these i's is substantially higher than $\frac{1}{2}$, according to the rules of statistics, then the reader gets a confirmation of the tag's identity.

Since the adversary may reactivate tags many times it is easy to collect a lot of data for cryptanalysis. However, the number of possible subsets is enormous even for relatively short identifiers, which makes brute force attacks extremely hard. On the other hand, noise introduced into tag's answers makes it problematic to start algebraic attacks (like the one from [32]).

Pseudo-Random Sequence

The idea presented in [9] is to share a pseudo-random sequence by the tag and the system running the tag. Namely, the system uses a pseudo-random keyed function F. Each tag T holds a unique key k_T and a couple of values computed with $F(\cdot, k_T)$. The system and the tag use pseudo-random numbers RN_i^T, where $RN_{i+1}^T = F\left(RN_i^T, k_T\right)$ for each i. The system and the tag should be synchronized. In a simplified version of the protocol both the tag and the system hold the same value RN_i^T. The tag authenticates itself with RN_i^T; the system response is RN_{i+1}^T. After this on both ends they switch the current string to RN_{i+2}^T, which can be computed with the secret key k_T.

The full version of the protocol is slightly more complicated as it has to handle the situations where the exchange gets interrupted, and the fact that the readers must not contain the secret k_T but only a few values RN_i^T in order to avoid breaking security of the system in case of corrupting a reader.

A similar idea is given in [50] for achieving transfer of authentication information of a tag from one organization to another so that afterward the first organization cannot trace the tag. The idea is that a tag holds a value t, while the owner holds (u, t) such that hash$(u) = t$. In order to authenticate a tag, the reader sends it a random string r_1; the tag chooses r_2 at random and computes $M_1 = t \oplus r_2$, $M_2 = f_t(r_1 \oplus r_2)$ and sends M_1, M_2 to the reader. The reader consults the system server, which computes r_2 as $M_1 \oplus t$ and checks if $M_2 = f_t(r_1 \oplus r_2)$. If they are equal, the tag is authenticated. In order to change the tag's identity t, the system sends it u masked with r_2 (shifted circularly). Then the new u is determined as $(u \ll l/4) \oplus (t \gg l/4) \oplus r_1 \oplus r_2$ and the tag stores the value $h(u)$ as the new t (l is the length of the strings used).

22.2.3.2 A Scalable Solution: Authentication of Travel Documents

Schemes based on symmetric methods are not easily scalable. If the number of objects is fixed and/or small they are quite attractive; for large-scale systems the readers have to handle huge amounts of data and perform nontrivial computations. Therefore, for big open systems, solutions based on asymmetric cryptography seem to be inevitable. A high-end example of this kind are protocols for communication with electronic identification documents [4].

The easiest way to establish communication with mobile devices is to use static Diffie–Hellman protocol. *Static* means here that the value $x_X = g^{r_A} \bmod p$ for unit A is fixed and used for establishing each session key. So x_A is the public key of A and can be signed by the system provider. The value r_A is the secret used by A exactly as for the usual Diffie–Hellman protocol (cf. TinyPK in Sect. 22.2.2.6).

Apart from establishing a shared session key, the protocol should enable authentication of the reader against the tag (Terminal Authentication). The architecture built for this purpose is a public key infrastructure with the public key of the root known to the mobile unit. The infrastructure provides each entitled reader a chain of certificates for a public key of this reader and statement of the reader's rights. The chain of certificates has the property that it can be verified with a single public key of the root. The certificate given for the reader should have a short validity period so that the mobile unit can be sure that the right statement is fresh.

The mobile unit performs two steps: first, it has to check the chain of certificates. Then it sends a random challenge to the reader, which has to answer with digital signature of the challenge signed with the private key corresponding to the key contained in the certificate. If the verification of the signature is positive, then the tag can assume that the reader holds the private key and therefore is the one certified by the infrastructure with the root as anchor point of trust.

Another important idea is to protect devices from being used without consent of the owner. According to the PACE protocol [4] activating a device A (knowing a password p) requires knowledge of p (it is a secret of the owner of A or an optically readible code). In the first step a random nonce s encrypted with the password p is sent to A.

Then a modified Diffie-Hellman key agreement is executed – a possible modification is that the agreed common secret $g^{x_a * x_B}$ is multiplied by g^s. A nice feature of this scheme is that s is hidden and therefore passive attacks are hard even for low entropy passwords.

22.2.4 Communication Authentication: A Framework Example

By design, wireless sensor networks are susceptible to distortions in messages passed among nodes. Partially it is caused by specifics of a radio channel, but here we shall focus on problem of false data generation and injection into the network by malicious nodes and attackers. Since networks usually work without human supervision, physical tampering with stations is practically possible, and it is relatively easy to eavesdrop and intervene with radio transmissions.

22.2.4.1 Multi-hop Group Authentication

For an energy-constrained network it is important in that an attacker may try to inject data aiming at congesting the entire network or depleting nodes' energy on message relaying. Hence limiting the area where malicious data propagate may spare the rest of the network.

A framework responding to these threats is presented in [57]. The goal is to authenticate information regarding some events detected by a cluster of sensors in a local area and contained in a single report sent to a base station. The origin of the report needs to be checked not only at the destination place, but also en route so that fake injected reports can be deleted early.

The network is initiated in so that it consists of clusters containing $t + 1$ nodes, with a cluster head (CH) in each cluster. For each cluster there is a fixed communication path to the base station (BS). In the following we consider a cluster C consisting of the nodes $u_1, \ldots, u_t, CH$ (CH being the cluster head) and a path $v_1, v_2, \ldots, v_k$ between CH and the base station. Within the framework we use shared symmetric keys. Namely, each node u holds a key K_u^{BS} shared with the base station, the keys to communicate within the cluster (and in particular with the cluster head), and the keys for authentication with "associated nodes." Specifically, the associated nodes of v_i are the nodes $t + 1$ steps ahead and $t + 1$ behind on the way to the base station. Additionally, v_{t+1} is associated with CH, and for $i \leq t$ node v_i is associated with u_i. For clarity, for a given node, UA is its associated node closer to the base station. Similarly, call LA the associated node further away from the base station.

If an event E occurs in the cluster C, then the cluster issues collectively a report on E together with a set of message authentication codes. The first of them is a compressed version of MACs computed with the keys shared with the base station:

$$\mathrm{XMAC}(E) = \bigoplus_{u \in C} \mathrm{MAC}\left(K_u^{BS}, E\right).$$

XMAC(E) is forwarded to the base station along the path, its verification against E is possible by the base station only. Additionally, each cluster node u contributes a so-called *pairwise* MAC, PMAC, used for en route authentication by its UA:

$$\mathrm{MAC}\left(K_u^{UA}, E\right).$$

The report sent to the base station contains event description E, XMAC(E), and $t + 1$ PMACs. The initial PMACs are collected by CH and then sent to the base station. Each station on the route checks exactly one PMAC—the one coming from its LA-associated node. If the MAC is correct, then the checked PMAC is removed and a new PMAC, to be checked by its UA, is inserted.

The design of the protocol allows that if *at most t* compromised nodes inject false information, the breach will be detected after no more than t hops. In fact, t compromised nodes can, at best, generate t consecutive valid PMACs that will check with t non-compromised nodes. However, the PMAC list contains $t+1$ elements, so at least one of them will contain a PMAC calculated for true data and its verification will fail. On the other hand, if the injection takes place at point less than t hops from base station, the verification will fail at calculating $\mathrm{XMAC}(E)$ at the base station. Similar argumentation passes for t compromised nodes in a cluster.

The proposed scheme again illustrates the concept of cooperation between sensors and resistance to injected data as an emergent property of the network. The calculations are spread evenly among the units and the overall outcome (i.e., data integrity) is probably greater than the summarized contributions of all the nodes acting by themselves could be.

22.3 Key Management

In many cases the usage of asymmetric cryptography for establishing secure links is prohibited by hardware limitations. In this case we have to depend on symmetric methods which require that the communicating parties share secret keys. The keys have to be preinstalled in the devices before they are deployed in the field, or there is an initialization protocol, such that the shared keys are initialized under controlled security conditions.

Predistribution of the keys might be a hard problem, since quite often we do not know in advance which pairs of stations will communicate. In particular, this is the case when devices are mobile and their routes are unknown in advance.

22.3.1 Master Key Schemes

If all devices are deployed by the same provider, then the simplest solution is to preload each of them with the same secret key. This method has however a strong drawback—an adversary only needs to capture a single device to break system security entirely.

If the nodes have fixed positions one can avoid most of these problems by using a single master key [16]. The idea is that the master secret K_m stored in each newly deployed node will be removed from the node immediately after establishing secure links with the neighbors. Namely, when the network is initiated, then the nodes i and j establish their shared key $K_{i,j} = F(K_m, i, j)$, where F should have properties of a cryptographic hash function. Therefore, each link should have a dedicated key and leaking one $K_{i,j}$ should not endanger other link keys. Of course, this method is limited to system initialization: if a new node s is installed, then no old node i can compute the key $K_{i,s}$, as K_m is already removed from the memory of node i. The solution to this problem is that node i retains a key $K_i = F(K_m, i)$. In this scenario the new node s computes the keys $K_{i,j}$ and K_i using K_m and sends $K_{i,j}$ encrypted with K_i to node i. Since node i holds the key K_i, it can recover $K_{i,j}$. On the other hand, if an adversary breaks into i and learns K_i, he can get automatically all the keys $K_{i,j}$. However, after breaking into K_i he can do it anyway. Moreover, knowledge of K_i does not directly endanger any other node. For more discussion about the scheme see also Sect. 22.3.7.

A more structured design for determining pairwise keys is the classical Blom's scheme [6]. It offers a separate key for every pair of devices that remains secure as long as less than λ devices are corrupted. The general idea of the scheme is as follows: if there are n devices, then the system provider chooses appropriate $(\lambda+1)\times n$ matrix G over a finite field with more than n elements. G is public. Then the provider chooses a secret $(\lambda + 1) \times (\lambda + 1)$ symmetric matrix D and computes $A = (D\cdot G)^T$. The device i gets the ith row of A; the pairwise keys are derived from the matrix $K = A \cdot G$. Let us observe that K is symmetric: $K = (D \cdot G)^T \cdot G = G^T \cdot D^T \cdot G = G^T \cdot D \cdot G = G^T \cdot A^T = (A \cdot G)^T = K^T$. The key shared by devices i and j is $K_{ij} = K_{ji}$. In order to find it, device i multiplies the row i of A by column j of G. Device j gets the same result by multiplying the row j of A by the column i of G.

22.3.2 Random Assignment Schemes

22.3.2.1 Basic Scheme

The idea from [20] is to generate a large pool of keys and to assign to each device its random subset. By the birthday paradox we know that if the key pool K has size n, then two random k-element subsets of K are disjoint with probability less than $\frac{1}{2}$ for k of order $\sqrt{n}$. In this scenario each device gets a random subset of k keys from the pool. If such devices want to establish communication, they use the key(s) shared by them for establishing the link key.

The problem with the basic scheme is that an adversary can still retrieve keys from the devices under his control. However, since each device holds only a limited number of keys, this does not lead to a complete breaking the system.

In order to increase immunity of the scheme against an adversary collecting the keys from the pool one can use q-composite schemes [12]. Then, in order to establish a link at least q shared keys are necessary. This seems to decrease the chances of the adversary: he has to own $\underline{\text{all}}$ shared keys used by communicating parties. On the other hand, the scheme has the drawback that the number of keys in each device must be higher (for sharing k keys the size of the random subsets needs to be at least of the order $n^{(k-1)/k}$). This in turn increases the efficiency of collecting the keys by the adversary: q-composite key is superior over the basic key only if the number of devices corrupted by the adversary is relatively low.

Another attempt to make the system more resilient to the adversary collecting the keys is to use multipath key establishment [2, 12]. The idea is that in order to establish a key between the nodes A and B we use some number of intermediate nodes, say $E_1, \dots, E_k$, where each E_i can connect both with A and B. Over each link A, E_i, B the nodes A and B establish a separate key, the final key for the link A, B is a combination of all these keys (for instance, the keys can be XOR-ed). If at least one of these links cannot be eavesdropped by the adversary, then the key established for the link A, B is secure.

The second important issue for random key predistribution is that randomized assignment of the keys implies that some pairs of nodes have more shared keys than average. On the other hand – for some pairs the number of shared keys is far lower than average. The way to reach a more regular system design is to use more structured subsets of the key pool. One idea of such a combinatorial design [11] is to assign the keys to points of a projective space. Then, each device gets a subset of keys that corresponds to a line in a projective space. Since each two lines in the space intersect we can be sure that two devices share a key.

One can also use hybrid techniques such as [17] that combines Blom's scheme and random predistribution. Furthermore, deterministic and quasi-random approaches based on combinatorial design theory were proposed [10]. Specifically, BIBD (*Balanced Incomplete Block Design*) problem and the concept of *finite generalized quadrangles* are employed to create adequate and effective (in terms of the size of key share) mappings from the space of key pool.

22.3.2.2 Exploiting Properties of the Communication Channel

In the work we mentioned before in Sect. 22.2.1, Miller and Vaida present a protocol that allows for efficient pairwise keys establishing in the network with the help of both powerful third party and some aspects of radio communication [40]. First, prior to deployment, the third party masterminds the security infrastructure of the future network at some level of generality. To this end, each sensor is given α distinct keys, a set that is not shared with any other node. Furthermore, a Bloom filter[3] containing all node's keys is calculated for each station and a Merkle Tree (MT) with ID's of nodes and the values of corresponding filters as leaves is computed. Next, each node is given its Bloom filter along with MT values needed to authenticate the filter.

Initialization phase: Immediately after deployment, each sensor broadcasts its Bloom filter and MT credentials to make them known to all its one-hop neighbors. This information is next used in the second stage, where all sensors switch to a randomly and uniformly chosen radio channel, listen to it, and then broadcast one of its keys *in plain text*. This procedure is repeated on different channels until all α keys have been transmitted. Upon hearing a key from its neighbor, the node confronts its value with the Bloom filter it received from this neighbor during the first stage and accepts the key only if it is a positive match. Due to randomness of the process, spatial diversity of nodes' location, and possible packet loss due to interference, at the end of this phase with high probability there will be at least one key shared between two nodes that will not be known to any third node.

Key discovery phase: After key broadcast is over, each node computes a new Bloom filter (B_d) hashing in all its α keys and some predetermined fraction of keys it heard during previous phase. The new filter is broadcast on a common channel. Upon receiving such filter from all its neighbors, the

[3] see footnote in Sect. 22.2.1.3

node can attempt to determine the keys it shares with each of them. To do that it chooses a set of ν keys that all belong to both its B_d filter and the filter it received from a neighbor and performs a hash on their concatenation, obtaining a candidate key K_c to be used in the last phase. When no ν common keys can be selected for a given neighbor, the node may decide to give up this link or request to re-run the initialization phase.

Key establishment phase: In the final phase the node challenges its neighbor with candidate key K_c. It sends a message containing a random nonce encrypted with K_c along with Bloom filter B_ν containing all ν keys contributing to K_c. If the neighbor is able to reconstruct the K_c using the information stored in B_ν it will answer with the same nonce incremented by one and encrypted with K_c, the now pairwise secret key. Since Bloom filters are liable to false positives, it is possible that either the node or the neighbor came across such key causing false positive during construction or verification of B_ν, respectively. Should it happen, they can re-attempt to establish the K_c by choosing another set of ν keys.

Using Bloom filters the sensors are able to effectively transmit information about their shared keys and authenticate them with Merkle Tree mechanism. More importantly, the resulting security (and privacy) of communication comes as a result of physical properties of the radio channel, and not only by the implicit mechanisms of the protocol.

22.3.3 Polynomial Share

In Sect. 22.2.2.3 we have seen how bi- and tri-variate polynomials can be used for authenticating messages. In [36] Liu and Ning provide descriptions for two instantiations of such framework: an extension of "basic scheme" pool-based predistribution and grid-based approach. Recall that we are interested in polynomials of degree t such that $f(x, y) = f(y, x)$. Then, the *polynomial share* of a node i is $f(i, y)$ and, once two nodes i and j have agreed on a common polynomial f, they can compute pairwise key to be $f(i, j) = f(j, i)$. With these assumptions, the key problem to be addressed is the way to establish a common polynomial.

In the first instantiation, the polynomial shares are distributed prior to deployment at random from a set of polynomials generated by a setup server. Next, the nodes attempt to directly establish keys with one another. That can be achieved either by using some pre-deployment information preloaded by the server or in real time. The downside of the former solution is that if an attacker captures a single node he can learn the polynomial shares for all its neighbors-to-be. This problem is mitigated in the latter solution at the expense of additional communication overhead. Here, nodes discover their common polynomials in an on-line fashion. For example, a node can blindly calculate the set of potential pairwise keys with its neighbors to be K_v and for all keys k broadcast pairs $\{\alpha, \mathrm{Enc}(\alpha)_k\}$, $k \in K_v$. If any other node can decrypt α with some key in its possession, it can reply to this broadcast to establish the common key. On the other hand, if no such decryption is possible, a node attempts to establish a path-key. To do that, the node will send out key establishment requests via the nodes it has already established pairwise keys with and the process will continue until the targeted node is reached.

In the second instantiation all sensors are mapped into a two-dimensional grid, and the server assigns two polynomial shares to each of them: one corresponding to the row, and another corresponding to the column in the grid where the node is located. To discover a common polynomial with some other node the station needs to verify if it is located on the same column or row in the grid and attempt to use corresponding polynomial to calculate a common key. If the nodes do not share either polynomial, they run a path-key establishment protocol querying possible intermediate nodes (ones at the crossing of adequate column and row). The exact mechanism employing this framework is explained in detail in Sect. 22.3.4. Here we stress only the cooperative nature of proposed mechanism, where a sensor network works out a particular feature (i.e., pairwise keys) based on limited information stored in each node. In essence, the proposed framework illustrates a common mechanism where a powerful server in pre-deployment stage preloads nodes with limited data, which then is combined into a coherent and complete knowledge in a cooperative system.

22.3.4 Multi-group Deployment

Design of key distribution scheme might be based on splitting the network nodes into groups, and building up the overall system on the subsystems designed for groups. An example of such an approach is presented by Liu et al. in [37]. The idea is that first we split the nodes into groups G_i based on their (approximate) location (so-called *deployment groups*). For each group, a *different* instance of key predistribution scheme $\mathcal{D}$ is run to establish keys, called *in-group* keys. Next, *cross-group* keys are established. Namely, the sensors are further subdivided into m cross-groups G'_i and for each such group a new, different instance of $\mathcal{D}$ is applied. The requirement on G'_i is that (1) each cross-group includes exactly one sensor from each deployment group and (2) no sensor belongs to more than one cross-group. One can imagine that the sensors are assigned to a grid, so that the nodes assigned to row i form group G_i and the nodes assigned to column j form group G'_j.

Simply put, each sensor is in possession of pairwise keys with all nodes in the same row and column, but in order to establish a communication with other nodes it needs to use one of the already associated nodes as proxy. This path is called *a bridge* and it can be built in two ways: either by querying a node in the same cross-group to establish connection using its in-group key, or using own in-group key to query other node in the same deployment group to use its cross-group key.

Yet another approach is based on recursive splitting into subgroups [33]. The idea is to split the set of nodes into two groups, say A and B. Then we assign master keys K_A and K_B for inter-group communication: K_A is stored in all nodes of A and K_B is stored in all nodes of B. Additionally, during the predistribution phase a node $A_i \in A$ gets a key $K_{A_i} = F(K_B, A_i)$ and node $B_j \in B$ gets a key $K_{B_j} = F(K_A, B_j)$ (again, F is a function having properties of a secure hash function). The communication link between A_i and B_j can be then secured by a key derived from K_{A_i} and K_{B_j}. Observe that A_i holds K_{A_i} and can compute K_{B_j} using K_A. Likewise, B_j holds K_{B_j} and can compute K_{A_i} using K_B. Moreover, no other node of the network has access to both K_{A_i} and K_{B_j}, so the communication between A_i and B_j remains private. In order to provide the keys for communication within A and B, we use exactly the same procedure recursively (and independently for each instance). Finally, the number of keys stored by a single device is $2\lceil \log n \rceil$, where n is the number of keys. This is much less than in the case of Bloom's scheme. On the downside, it suffices to break into a single device of group A and a single device of group B to get full access to any communication between a node in A and a node in B.

Similar ideas are presented in [19]. The problem addressed is adjusting the key assignment to particular communication graph architectures. Tailored solutions for diverse families of graphs (e.g., planar graphs, bipartite graphs) are presented.

22.3.5 Powerful Third Party

By introducing a slightly different model of a network, Traynor et al. discuss the issues of security in heterogeneous networks of sensors [51]. In such a network there are two kinds of nodes: the small sensory stations (L1 level) with limited capabilities performing data collection tasks, and the more powerful ones (L2) able to perform some effective encryption algorithms and equipped with a better radio. In their scheme all the nodes follow the basic scheme [20], with the difference that nodes in level L2 store bigger subset of keys from the pool than these in L1. It naturally follows that more powerful (and secure) nodes have better connectivity in the network, which supports the three following modes of trusted communication:

Backhaul trust —when L1 nodes simply transmit information to an external sink through their nearest L2 neighbor; this imposes trust requirements only on one particular L2 node as the entire second layer can be considered secure and well connected.

L1 limited trust —for transmissions within L1 layer across different neighborhoods; to this end either the two L1 nodes share a common key as a result of the basic scheme or can establish a session key via a path of L2 nodes.

L1 liberal trust —for similar case as above, but where session key is agreed along a path within L1 layer only; this increases the risk of losing secrecy as any intermediate node can leak information when captured, and potentially many nodes involved in the process increase this threat.

Two schemes are presented in [51]. First, LION, for stand-alone key management in a system without a central trusted station. Second, TIGER, leveraging the presence of a Key Distribution Center (KDC) to facilitate pairwise key establishment. LIGER, a merge of the two, enables the network to maintain its functionality both with and without the KDC.

LION. This protocol is an extension of the basic scheme described in [20], implementing the concept of limited trust as described above. First, an L1 node learns its neighborhood and discovers the keys shared with the nodes therein. For all nodes in the neighborhood not sharing a key with the given node a *request for assistance* message is sent to the nearby L2 nodes, requesting that pairwise keys with these L1 nodes be established with L2 node acting as a proxy. Finally, all nodes participating in establishing such keys change or delete the keying material involved in this operation so that should a single node be compromised, it does not reveal any other secrets.

TIGER. The nodes are preloaded with a subset of the keys from a pool as in the basic scheme. Additionally, L2 nodes can perform public key authentication with the KDC. Also, each L1 node creates an *authenticator key* as a result of XOR-ing a subset of its key share, and a bitmap representing the keys used in this operation. Node A, to request a pairwise key establishment with a peer node in L1, say B, creates a token containing its authenticator and the bitmap. Node B generates its own token and forwards both tokens through L2 to the KDC. The KDC can verify, based on the bitmaps, the validity of authenticator keys and therefore confirm authenticity of both nodes. Then, the KDC generates new authenticator keys k_A, k_B and adequate bitmaps b_A, b_B for A and B, respectively, and randomly chooses a pairwise key K_{AB} for them. Lastly, it encrypts the key with the new authenticator keys and sends back pairs: $(\text{Enc}(K_{AB})_{k_A}, b_A)$ to A and $(\text{Enc}(K_{AB})_{k_B}, b_B)$ to B. The nodes can then regenerate authenticator keys and extract K_{AB}.

In a similar way, L1–L2 nodes authentication is performed. On request, the L1 node issues a token to L2. The token is forwarded to KDC to confirm the authenticity of the L1 node. KDC establishes a session key and sends it to L2 along with its copy encrypted with the L1's authenticator key. L2 forwards this message to L1 where the session key can be recovered, and hence L2 authenticated to L1. In essence, TIGER is a lightweight version of Kerberos (see e.g., [30]) tailored to the needs of sensor network in that the number of exchanged messages is limited and encryption is modified for less demanding devices. Both presented schemes utilize the same pool of keys which saves storage on constrained nodes and provides seamless transition between the two scenarios.

22.3.6 Dynamic Key Structures

22.3.6.1 Key Evolution

If a device is not under continuous supervision and is not tamper resistant, then an adversary may get access to it, read the keys from the internal memory, and then let the device work as before. Even the strongest key assignment and encryption/authentication methods do not provide any protection against such attacks.

One costly method to cope with this problem is to provide tamper-resistant or tamper-evident devices. However, there is an alternative way due to Ren, Tanmoy, and Zhou based on key evolution [47] for securing pairwise keys. The idea for managing a key for securing the link between devices A and B is as follows:

- A and B initialize their shared symmetric key K using any method available;
- each time when devices A and B communicate, they change slightly the key K in a random way.

Moreover, the changes of the key do not involve any additional communication. In more detail, if A sends a message M to B, then the following steps are executed:

1. A chooses at random a bit in K and flips it. It stores K' as the shared key for communication with B. Then A encrypts M with the new key K' and sends the resulting ciphertext C to B.
2. B receives C and performs trial decryption with all keys that can be derived from K by flipping a single bit. Only the result for key K' turns out to be valid, so B can recognize that the key has been changed to K'. B replaces K by K'.

In fact, the protocol is slightly more complicated to keep the changes of the shared key synchronized in the case when some messages get lost (see [47] for details).

The main properties of the protocol are the following:

- Coordination of key changes requires no additional communication, the overhead is on computation. This is motivated by the fact that communication consumes much more energy and time than computation.
- The solution is *forward-secure*. Namely, if the adversary breaks into device A at time t and gets the shared key K, but later does not observe some number of messages exchanged between A and B, then the adversary loses his knowledge about the shared key. Indeed, each communication changes one of the bits at random. So after some number of steps the key changes completely.

A single drawback is that the scheme is not backward-secure: if the adversary records all communication and then breaks into A, then the recorded communication can be decrypted by reversing the key changes step by step. However, as observed in [28], we can get backward-security if we adopt the following key evolution scheme:

$$K' := F(K, i, t)$$

where F is a pseudo-random one-way function, i is a parameter chosen at random form a small range, and t is the change number. In this case, even if the adversary has the key K', it is implausible (with minor assumptions regarding $F(\cdot)$) to derive the previous key.

22.3.6.2 Key Levels

Hash chains can be used in a yet another way in order to improve resilience against capturing nodes and compromising the keys assigned via key predistribution schemes. According to [13] assume that each key K_i from the pool of keys appears in n different versions $K_1, K_2, \ldots, K_n$, where

$$K_1 = K \quad \text{and} \quad K_{i+1} = F(K_i) \quad \text{for } i = 2, 3, \ldots$$

and F is a one-way function. In place where the original predistribution scheme assigns the key K, the scheme with levels chooses one of the levels i and assigns K_i. Observe that if two devices have to use the shared K, in the modified scheme they hold some K_i and K_j and both can use $K_{\max\{i,j\}}$. Simply, if $i < j$, then the device holding K_i can derive K_j as $F^{j-i}(K_i)$. So, the only cost is a certain increase in local computation.

The point is that in the original scheme if the adversary corrupts a single device holding K, then K becomes useless. In the scheme with key levels with some luck the adversary corrupts a device holding K_t, but the communicating devices hold K_i, K_j, where $i, j < t$. Then the adversary is still unable to decrypt the communication. Unfortunately, it turns out that the probability of this event is at most $\frac{1}{3}$. On the other hand, it is close to $\frac{1}{3}$ already for 10 levels. If we use multiple keys to establish a connection and two level schemes, then the expected number of devices that we have to break into increases at most by 50%.

22.3.6.3 Dynamic Key Management with Key Levels

Assume that a provider supports mobile devices using random key predistribution and that the devices may be updated at kiosks which they visit from time to time. One can apply key levels to keep the system immune against an adversary capturing devices and retrieving keys from them.

Each key K has infinitely many levels K_i, for $i = 1, 0, -1, \ldots$. The system provider keeps a trapdoor to F and therefore can compute K_{i-1} from K_i. On the other hand, a user device (or a kiosk) holding K_i can derive only keys K_j for $j > i$. The keys are uploaded to the kiosks periodically, each time decreasing the current level. If a device has not updated the keys recently, the probability that the keys it holds are already known by the adversary increases. So any other device may decide whether the key levels supported by such communication partner are fresh enough to offer sufficient level of security.

22.3.7 LEAP: A Full Key Infrastructure

Even in relatively simple scenarios, there are different types of messages exchanged in a sensor network. Some are local by nature, e.g., neighbor discovery packets, some are spread globally to all (broadcast) or sent to specific nodes (node-to-node communication). This observation along with conclusion that different mechanisms are best suited for securing different communications is at the basis of LEAP (Localized Encryption and Authentication Protocol) [56]. The basic building block for key generation in LEAP is a family pseudo-random functions $\{f_k(\cdot)\}$, parameterized with some key k. In implementation, the RC5 family was used for both key generation and MAC computation, which resulted in better code reuse and implementation size reduction. Below are details on each type of keys proposed by LEAP.

Individual key: This is a key shared by a node with the base station. For node u the key is $K_u^m = f_{K^m}(u)$, where K^m is the secret master key of the network controller. Note that there is no need for the controller to store all nodes' keys as they can be generated on-the-fly. K_u^m is put into node's memory before deployment.

Pairwise shared keys: Most communication happens locally and should be protected by pairwise shared keys. A natural scenario for establishing such key would be that a newly deployed node wishes to gain access to the network and gain necessary credentials for further operation. However, an attacker can capture the newly arriving node before it manages to receive credentials and use the node's secrets to gain access to the network. The protocol assumes that such an attack requires at least time $T_{\min}$. The pairwise keys are generated as follows:

- in the pre-deployment phase, a common key K_I is stored in every node's memory; each node u can then compute its master key: $K_u = f_{K_I}(u)$;
- on deployment, node u sets a timer to fire after $T_{\min}$;
- node u sends out a HELLO message; on receipt, each node v responds with an ACK in the form $\{v, \mathrm{MAC}(K_v, u, v)\}$; since u knows v's ID and (still) it knows K_I, it can compute K_v and use it to verify the ACK;
- node u then computes its pairwise key with v to be $K_{uv} = f_{K_v}(u)$; v performs the same operation; note that no additional exchange of messages is required; all consecutive communication using K_{uv} will authenticate both u and v;
- when timer fires, the node deletes permanently K_I and all master keys K_v of its neighbors; it keeps however its own master key K_u.

Note that when a new node arrives, it can join the network using K_I, while the nodes already present in the network do not need K_I to issue ACK. Also, even if an adversary captures one node, the only information that leaks is that of the node's keys with its neighbors, and no general secret information can be extracted. Furthermore, in an event of a cloning attack, the credentials stored in a duplicated node are not sufficient to establish pairwise keys with other nodes; hence planting the duplicate in a remote location is thwarted. Finally, an extension of the above protocol is provided to cope with the case when K_I is compromised. In essence, the idea is to use a sequence of initial keys to be used at each time interval wherein a new node arrives, rather than one global K_I. The proposed approach provides both forward and backward confidentiality (see [56]).

Cluster keys: One common key shared by all nodes in a cluster enables in-network processing and, more broadly, an insight into messages transmitted within the cluster, while at the same time keeping it hidden from nodes in other clusters. If a forwarding node can decrypt a, say, "cluster-MAX-query" message, it can decide whether its own readings contribute to the reported value or not and save energy on computations. Establishing a cluster key is straightforward in that a sensor u generates a random cluster key K_u^c and sends it to all nodes it wishes to, using the appropriate pairwise keys.

Group keys: A group key K_g is a key shared by all nodes in the network used to secure network-wide communication. While the first group key may be loaded to sensors at deployment, the essential case here is *re-keying*, as prolonged use of the same key makes some attacks more feasible. The simplest solution would be for all nodes to perform substitution: $K_g := f_{K_g}(0)$ on timely basis, given that time synchronization can be achieved. More importantly, the group key needs to be changed whenever some node gets revoked as malicious. To invoke the procedure, the controller (base station) broadcasts a revocation message

$$\left\langle u, f_{K_g'}(0), MAC\left(k_i^T, u, f_{K_g'}(0)\right)\right\rangle$$

where u is the revoked node ID, K_g' the new group key, and k_i^T the to-be-revealed hash chain value (cf. Sect. 22.2.1.2). As in μTesla, k_i^T is revealed after certain time and can be used to verify the authenticity of revocation message. If it succeeds, $f_{K_g'}(0)$ is stored for further use and the second stage commences. This basically uses breadth-first message passing, originating at the base station. The base station informs all nodes in its cluster about the new group key K_g', securing the communication with its cluster key. The informed nodes pass the information further using their own cluster keys. On receiving the new key, a node can compute $f_{K_g'}(0)$ and verify it with the value obtained from the revocation message.

22.4 Encoding

Once the nodes of the network share symmetric keys, one can use them for securing transmissions, in the sense of confidentiality (encryption) and integrity (message authentication codes, packet counters). The technique that we discuss here concerns appropriate message encoding that minimizes risk of information leakage.

22.4.1 Multiple Paths

If a data stream is sent via a single path, then the adversary has to compromise only a single node on the path in order to break security of the transmission completely. A simple countermeasure is to split messages into two parts and sent them through different paths. For instance, if A sends a message M to B, then it encrypts M

with a key K shared with B, $C = E_K(M)$, and splits C into C_1 and C_2. Namely, C_1 contains all odd bits of C and C_2 contains all even bits of C. Then C_1 and C_2 are sent via separate paths. On each hop of such a path, C_i is additionally encrypted with the key shared by the sender and the receiver. Observe that even if the adversary learns some C_i, it cannot even start standard cryptanalysis, since only half of the ciphertext is available.

The approach with two different paths has an additional advantage for typical security applications: if the nodes are deployed in the field, it is much harder to find two nodes (say in the grass), than only one. This technique is announced in [31]: a path from A to B is chosen such that it goes through pairs of intermediate nodes, say $(P_1, P'_1), (P_2, P'_2), \ldots, (P_m, P'_m)$. The nodes P_i and P'_i receive, respectively, C_i and C'_i from P_{i-1} and P'_{i-1}, with $C = C_i \oplus C'_i$. Then P_i splits C_i at random into $\underline{C_i}$ and $\overline{C_i}$, $C_i = \underline{C_i} \oplus \overline{C_i}$, and sends them over encrypted links to, respectively, P_{i+1} and P'_{i+1}. Similarly, P'_i splits C'_i at random into $\underline{C'_i}$ and $\overline{C'_i}$ and sends them over encrypted links to, respectively, P_{i+1} and P'_{i+1}. Then P_{i+1} computes $C_{i+1} = \underline{C_i} \oplus \underline{C'_i}$, while P'_{i+1} computes $C'_{i+1} = \overline{C_i} \oplus \overline{C'_i}$. Note that if we apply the method of two parallel paths described before, then compromising any two nodes of different paths reveals C. For the last method it would be necessary to compromise two nodes with the same distance from the destination.

22.4.2 Block Ciphers

Block ciphers employ a symmetric key shared by both communicating parties in order to provide both encryption and decryption. The name comes from the way they operate on data: it is split into blocks of fixed length and each part undergoes the same modifications. The key can be used as *initial vector* or as a parameter controlling these operations. The two widely known and used block ciphers are DES (Data Encryption Standard) and AES (Advanced Encryption Standard). These two however are problematic for use in sensor motes due to their relatively high demand for memory and calculations. Below we present some recent protocols implementing block ciphers schemes that fit better in such constrained hardware.

The first one, MiniSec [38], proposed by Luk et al. uses an OCB (Offset Code Book) mode of operation [48] with Skipjack as the underlying cipher. The OCB allows *both* encryption and authentication of data with very little overhead in packet size and only a single scan over the plain text. That is, for a message M of length $|M|$ the resulting codeword at the output has length $|M| + \tau$, and the probability of forging a valid message is $2^{-\tau}$. Generation of a ciphertext $\mathcal{C}$ is parameterized by a nonce N and a symmetric key K, whereas the resulting *tag*, authenticating the message is concurrently calculated as a function of $\mathcal{C}$, M, and an optional header. More precisely, during encryption the algorithm uses an auxiliary value Δ, which chains the ciphertexts of the blocks together. It is initialized as $E_K(N)$. For each block the value of Δ is updated by shifting it left by one position $\Delta := \Delta \ll 1$, if the most significant bit of Δ is 0, or as

$\Delta := (\Delta \ll 1) \oplus 0x00000000000000000000000000000087$, if this bit of Δ is 1. For encryption of a block we use the formula:

$$C_i = \Delta \oplus E_K(M_i \oplus \Delta)$$

Slightly different rules are applied when constructing the last ciphertext block that holds the tag.

In MiniSec each pair of nodes shares two secret keys, each for securing communication in one way. Additionally, for each such key an internal counter is kept and for each use of the key incremented by one. For encryption, adequate secret key is used and the corresponding counter value set for nonce N. For decryption, the receiver uses its own counter value for the key used in encryption and increments it afterward. As in many algorithms presented so far, the communicating parties can get desynchronized. A simple countermeasure is that the last x bits of the counter are sent along with the message (in the header). By piggybacking this information with the packet, MiniSec achieves counter synchronization for free: the receiver can temporarily update its counter according to the value in the packet and attempt to decrypt/authenticate it with the new value. If it fails, the counter is rolled back, while it gets synchronized in opposite case. In normal circumstances, it suffices to take small x, so the bandwidth overhead is low.

For broadcast messages usage of counters is problematic: since this is not a one-to-one transmission, each node would have to store a large set of counters for each potential broadcasting source, which, to begin with, may be hard to define. The authors propose two solutions. First, to use loose time synchronization, as the one in μTesla ([46]) and split the time into epochs. The epoch number during which communication takes place is used as the nonce N. This brings the threat of an attacker being able to forge a valid message throughout the same epoch. This drawback is addressed in the second solution. Here the sender S with identifier ID keeps an internal counter C_S of packets sent in the current epoch. This is short enough to be sent along with the message and pose no significant overhead. Then, in epoch E_i the nonce $N = ID \| C_S \| E_i$ is used for OCB encoding. On reception, the node checks the validity of the packet using the supplied value C_S. Additionally, to prevent playback attacks it stores the value $C_S \| ID$ in a Bloom filter ([7]) or rises an alarm when the filter already contains it. Due to small confusion caused by splitting time into epochs, two filters need to be kept for two consecutive epochs, but their size is relatively small and depends on the number of messages sent per epoch.

An interesting attempt to capture the multitude of devices using wireless communication (and as such, wireless sensor networks) under one hood is ZigBee Alliance [42]. This is a complete communication framework, and one of its building blocks include communication security by encryption and authentication. In fact, the proposed solutions are similar to the ones used in TinySec with only slight differences. ZigBee uses AES-128 as the underlying block cipher protocol, run in CCM* mode [18], offering both authentication and encryption, or either of the two. Both communicating parties are required to be in possession of a secret key and keep the associated counter for each key and security level. On reception, the local value of the counter is compared with the one transmitted, and the whole message is discarded when the local value is greater. This mechanism is used to prevent replay attacks (e.g., in DoS attempt) that can be performed with the same counter value. Additionally, the counter value concatenated with source's identification sequence and security level parameter constitutes a nonce N that is used as one of the input parameters to the CCM* algorithm.

The implementation details can be found in [42], while here it is worth mentioning that as is, the ZigBee framework could be found problematic to implement in some standard sensor node hardware. This is mainly due to the use of AES block cipher with relatively long (128 bits) key, as well as additional transmission overhead. On the other hand, some interesting mechanisms are built in the ZigBee. For example, after successful verification of the packet, the receiver checks the status of the sending node on its internal *NeighbourhoodList*, and updates the entry to reflect the level of authorization or "trust" for this device. The good idea behind this is that such trust building is done at no additional cost (except for storage) and can be used in other protocols during normal operation.

To conclude this section we refer the reader to [34], where a comprehensive evaluation of block ciphers for use in wireless sensor networks can be found. Law et al. present there a systematic framework for assessing different aspects including such important factors as energy consumption, timing, and storage efficiency.

22.5 Compromised Node Detection

By design, wireless sensor networks are an effective tool for data gathering in a decentralized way. However, the interactions between single nodes are strictly localized, as the nodes (usually) do not move. Despite this local interconnections such networks remind more of a distributed system rather than a heterogeneous network of autonomic devices. That is, data collection is performed in a collective manner and the "welfare" of the network does not depend strictly on a single node, given that adequate routing algorithms are running. This regularity in the network, when viewed from datacentric point of view, may provide valuable information on misbehaving nodes. Next, the information can be used to raise an alarm and implement adequate measures against such units.

22.5.1 Alert-Based Protocols

In [54] a general model of alert-based protocols is given. The authors make no assumption on the way malicious nodes behave, instead, they introduce a number of idealized models for their framework. These include *sensor behavior model*, *observer model*, and an *identification function*. The network itself is also mapped to an idealized *observability graph*, wherein an edge between two nodes exists if and only if these two nodes can observe each others' activity. The role of an observer is to report an *alert* whenever it sees abnormal activity in one of its neighbors. To reflect different physical properties of network operation (e.g., sleep mode, interference) the decision on reporting the alert is parameterized by a probabilistic function of observer's reliability, accuracy, and effectiveness.

The alerts are collected by a base station and if, during a given time window, their number for a particular edge in the network graph exceeds a preset threshold, this edge is classified as *abnormal*. The next step is to construct an induced graph from the observability graph, containing (1) all abnormal edges and (2) edges between such two nodes that have a common neighbor connected with *exactly one* of them by an abnormal edge. The later option corresponds to the case of a triangle ABC, where the edge AB is abnormal, but AC and BC are not. It is easy to see that in this case at least two of nodes A, B, C must be compromised. Indeed, if say A is compromised and B is not, then C must be cheating too by not reporting the problems with A. If both A and B are compromised, then they may behave correctly against C, but send reports concerning the edge AB in order to pretend that the first case occurs. Lastly, an algorithm detecting a minimal vertex cover on the induced graph selects these nodes that are compromised, taking a security parameter K to be the maximal number of compromised nodes. The complexity of this last step is $\mathcal{O}\left(mn\sqrt{n}\right)$, where n is the number of vertices and m the number of edges in the inferred graph.

The intuition behind this framework is that one sensor node can have a good insight into what its neighbors are doing due to locality property of the network and sensed data. Reporting abnormal activity can signify both the reported or reporting sensor's misbehavior; therefore both units are "shortlisted" for future investigation.

22.5.2 Detect and Tolerate

One of the attacks possible in sensor networks is the replication attack. The solution to discovery of replicated nodes is by means of centralized comparison of all nodes' neighbor lists and localizations. This however brings considerable communication overhead, as whole lists from each node need to be transmitted. Alternatively, one can run a localized version of the above algorithm, thus reducing communication overhead. However, this algorithm will not detect replicas planted by the adversary in distant enough locations. Parno et al. in [43] present two protocols that allow detection of replicas in a network and, consequently, their effective revocation. The following steps describe the first approach, the *Randomized Multicast*:

1. each node α sends to each of its d neighbors a location claim containing its identifier, locator (e.g., from a GPS unit), and a valid signature of the claim:

$$\langle ID_\alpha, l_\alpha, Sign\,(H(ID_\alpha, l_\alpha))\rangle$$

2. each of the neighbors checks the signature and plausibility of l_α;
3. with probability p each neighbor chooses g different locations in the network and forwards the location claim to the nodes in the proximity of these locations;

4. each node that receives the forwarded claim becomes a *witness* for its issuer;
5. if a witness receives two different claims l_α and l'_α for the same ID_α it floods the network with l_α, l'_α, thus informing about the discovered duplicate.

While it provides high probability p of detecting multiple replicas, the algorithm poses huge storage overhead to the witnesses, as they need to store on average $p \cdot d \cdot g$ claims, which induce $\mathcal{O}(n^2)$ communication costs. This drawback is fixed in the second protocol, the *Line-Selected Multicast*, where each location claim is sent to proximity of r locations chosen at random. The improvement here relies on the fact that in a multi-hop network packets follow a path and all nodes relying the packet can store it. This draws virtual lines across the network plane and everywhere the lines intersect, a corresponding node may perform cross-check for duplicate claims. Very few lines (i.e., duplicate ID's claims "arriving" from different points in the network) are needed for the intersection to happen. In the complexity analysis, letting r be small value independent of n (which still makes the intersection probability high) leaves the communication complexity $\mathcal{O}\left(\sqrt{n}\right)$.

22.5.3 Suicidal Pointer

The aforementioned concept of voting, or any reputation-based system of punishing misbehaving nodes suffers from one, difficult to avoid problem. The misbehaving nodes can actively take part in voting or reputation building protocols. If such node can perform a Sybil attack, or more nodes collude, any voting can be taken over by the attacker. Moreover, false accusations can be set against legitimate nodes, e.g., in order to divert traffic routes from them and pass it through a node in possession of the attacker. Last but not least, as already seen, the protocols designed to countermeasure misbehavior are time and message consuming.

A simple and worthwhile solution is presented in [15]. There, a revocation decision is made by a *single* node. The cost of denunciation is honest node's suicide, i.e., the misbehaving and the honest nodes are treated the same way by the network, be it timely revocation, mutual session keys deletion, or permanent blacklisting. High as the cost may seem, an inherent property of sensor network is here used, namely high redundancy. Application of such solution should be balanced against possible merits of using a network without a single sensor but without potential intruders. Additionally, this protocol defends itself in that it is simple, provides almost no communication overhead, and runs independently of other nodes.

References

1. M. Ajtai. Generating hard instances of lattice problems (extended abstract). In: *STOC '96: Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, ACM, New York, NY, pages 99–108, 1996.
2. R. Anderson, Haowen Chan, and A. Perrig. Key infection: Smart trust for smart dust. In: *Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference*, pages 206–215 (October 2004).

3. L. Batina, N. Mentens, K. Sakiyama, B. Preneel, and I. Verbauwhede. Lowcost elliptic curve cryptography for wireless sensor networks. In Buttyán L., Gligor V. D., and Westhoff D., editors. ESAS, volume 4357 of *Lecture Notes in Computer Science*. Springer, Berlin, pages 6–17, 2006.
4. J. Bender, M. Fischlin, and D. Kügler. Security analysis of the pace key-agreement protocol. In Samarati P., Yung M., Martinelli F., and Ardagna C. A., editors. ISC, volume 5735 of *Lecture Notes in Computer Science*. Springer, Berlin, pages 33–48, 2009.
5. E.-O. Blaß and M. Zitterbart. Efficient Implementation of Elliptic Curve Cryptography for Wireless Sensor Networks. Telematics Technical Reports TM-2005-1 (March 2005).
6. Rolf Blom. An optimal class of symmetric key generation systems. In: *Eurocrypt*, pages 335–338, 1984.
7. B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13:422–426, 1970.
8. B. Bruhadeshwar, K. Kothapalli, and M. S. Deepya. Reducing the cost of session key establishment. In: *ARES*. IEEE Computer Society, Los Alamitos, CA, pages 369–373, 2009.
9. M. Burmester, B. de Medeiros, J. Munilla, and A. Peinado. Secure epc gen2 compliant radio frequency identification. In Ruiz P. M. and Garcia-Luna-Aceves J. J., editors. ADHOC-NOW, volume 5793 of *Lecture Notes in Computer Science*. Springer, Berlin, pages 227–240, 2009.
10. S. A. Çamtepe and B. Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. *IEEE/ACM Transaction on Networks*, 15(2):346–358, 2007.
11. S. A. Çamtepe and B. Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. In Samarati P., Ryan P. Y. A., Gollmann D., and Molva R., editors. ESORICS, volume 3193 of *Lecture Notes in Computer Science*. Springer, Berlin, pages 293–308, 2004.
12. H. Chan, A. Perrig, D. X. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*. IEEE Computer Society, Paris, pages 197–213, 2003.
13. J. Cichoń, J. Grząślewicz, and M. Kutyłowski. Forward-secure key evolution in wireless sensor networks. In *Algosensors*, 2009.
14. J. Cichoń, M. Klonowski, and M. Kutyłowski. Privacy protection for rfid with hidden subset identifiers. In Indulska J., Patterson D. J., Rodden T., and Ott M., editors. Pervasive, volume 5013 of *Lecture Notes in Computer Science*, Springer, Berlin, pages 298–314, 2008.
15. J. Clulow and T. Moore. Suicide for the common good: a new strategy for credential revocation in self-organizing systems. *SIGOPS Operating Systems Review*, 40(3):18–21, 2006.
16. T. Dimitriou. Securing communication trees in sensor networks. In Nikoletseas S. José E., and Rolim D. P., editors. Algosensors, volume 4240 of *Lecture Notes in Computer Science*. Springer, Berlin, pages 47–58, 2006.
17. W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key predistribution scheme for wireless sensor networks. In: *CCS '03: Proceedings of the 10th ACM Conference on Computer and Communications Security*. ACM, New York, NY, pages 42–51, 2003.
18. M. Dworkin, National Institute of Standards, and Technology (U.S.). Recommendation for block cipher modes of operation [electronic resource]: The CCM mode for authentication and confidentiality/Morris Dworkin. U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology, Gaithersburg, MD, 2004.
19. E. S. Elmallah, M. G. Gouda, and S. S. Kulkarni. Logarithmic keying. *TAAS*, 3(4), 2008.
20. L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In Atluri V., editor. *ACM Conference on Computer and Communications Security*, ACM, New York, NY, pages 41–47 (2002).
21. G. Gaubatz, J.-P. Kaps, E. Ozturk, and B. Sunar. State of the art in ultra-low power public key cryptography for wireless sensor networks. In: *PERCOMW '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*, IEEE Computer Society, Washington, DC, pages 146–150, 2005.
22. H. Gilbert, M. J. B. Robshaw, and Y. Seurin. Hb#: Increasing the security and efficiency of hb+. In Smart N. P., editor. Eurocrypt, volume 4965 of *Lecture Notes in Computer Science*. Springer, Berlin, pages 361–378, 2008.

23. T. Heer, S. Götz, O. G. Morchon, and K. Wehrle. Alpha: An adaptive and lightweight protocol for hop-by-hop authentication. In *CONEXT '08: Proceedings of the 2008 ACM CoNEXT Conference*, ACM, New York, NY, pages 1–12, 2008.
24. J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte. Ntrusign: Digital signatures using the ntru lattice. In Joye M., editor. CT-RSA, volume 2612 of *Lecture Notes in Computer Science*. Springer, Berlin, pages 122–140, 2003.
25. J. Hoffstein, J. Pipher, and J. H. Silverman. Ntru: A ring-based public key cryptosystem. In Buhler J., editor. ANTS, volume 1423 of *Lecture Notes in Computer Science*. Springer, Berlin, pages 267–288, 1998.
26. N. J. Hopper, and M. Blum. Secure human identification protocols. In Boyd C., editor. Asiacrypt, volume 2248 of *Lecture Notes in Computer Science*. Springer, Berlin, pages 52–66, 2001.
27. W. Hu, P. I. Corke, W. C. Shih, and L. Overs. secfleck: A public key technology platform for wireless sensor networks. In Roedig U. and Sreenan C. J., editors. EWSN, volume 5432 of *Lecture Notes in Computer Science*. Springer, Berlin, pages 296–311, 2009.
28. M. Klonowski, M. Kutyłowski, M. Ren, and K. Rybarczyk. Forwardsecure key evolution in wireless sensor networks. In Bao F., Ling S., Okamoto T., Wang H., and Xing C., editors. CANS, volume 4856 of *Lecture Notes in Computer Science*. Springer, Berlin, pages 102–120, 2007.
29. N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.
30. J. Kohl, F. J. Kohl, C. Neuman, and J. Steiner. Kerberos version 5 rfc, draft 3, 1990.
31. M. Koza, M. Klonowski, and M. Kutyłowski. Security challenges for wireless sensor networks. Dynamic routing as a security paradigm. *ERCIM News*, 76, 2009.
32. M. Krause and D. Stegemann. More on the security of linear rfid authentication protocols. In Jacobson Jr. M. J., Rijmen V., and Safavi-Naini R., editors. Selected areas in cryptography, volume 5867 of *Lecture Notes in Computer Science*. Springer, Berlin, pages 182–196, 2009.
33. S. Kulkarni, B. Bezawada, and M. G. Gouda. Optimal key distribution for secure communication. University of Texas, Austin MSU-CSE-07-189 (July 2007).
34. Y. W. Law, J. Doumen, and P. Hartel. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Transaction on Sensor Networks*, 2(1):65–93, 2006.
35. A. Liu and P. Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *IPSN '08: Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, IEEE Computer Society, Washington, DC, pages 245–256, 2008.
36. D. Liu, P. Ning. Establishing pairwise keys in distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM Conference on Computer and Communications Security*. ACM, New York, NY, pages 52–61, 2003.
37. D. Liu, P. Ning, and W. Du. Group-based key pre-distribution in wireless sensor networks. In *WiSe '05: Proceedings of the 4th ACM Workshop on Wireless Security*. ACM, New York, NY, pages 11–20, 2005.
38. M. Luk, G. Mezzour, A. Perrig, and V. Gligor. Minisec: A secure sensor network communication architecture. In *IPSN '07: Proceedings of the 6th International Conference on Information Processing in Sensor Networks*. ACM, New York, NY, pages 479–488, 2007.
39. D. Micciancio and S. Goldwasser. Complexity of Lattice Problems. Kluwer, Norwell, MA, (2002).
40. M. J. Miller and N. H. Vaidya. Leveraging channel diversity for key establishment in wireless sensor networks. In: *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12 (April 2006).
41. V. S. Miller. Use of elliptic curves in cryptography. In: *CRYPTO '85: Advances in Cryptology*. Springer, London, pages 417–426, 1986.
42. ZigBee Standards Organization. Zigbee specification document 053474r17. Specification Document 053474r17, ZigBee Alliance (January 2007).

43. B. Parno, A. Perrig, and V. D. Gligor. Distributed detection of node replication attacks in sensor networks. In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, Washington, DC, pages 49–63, 2005.
44. A. Perrig, R. Canetti, D. Tygar, and D. Song. The Tesla broadcast authentication protocol, 2002.
45. A. Perrig, R. Canetti, D. X. Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In: *NDSS*. The Internet Society, Reston, VA, 2001.
46. A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. Spins: Security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002.
47. M. Ren, T. K. Das, and J. Zhou. Diverging keys in wireless sensor networks. In Katsikas S. K., Lopez J., Backes M., Gritzalis S., and Preneel B., editors. ISC, volume 4176 of *Lecture Notes in Computer Science*. Springer, Berlin, pages 257–269, 2006.
48. P. Rogaway, M. Bellare, and J. Black. Ocb: A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information and Systems Security*, 6(3):365–403, 2003.
49. R. Roman, C. Alcaraz, and J. Lopez. A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes. *Mobile Networks and Applications*, 12(4):231–244, 2007.
50. B. Song and C. J. Mitchell. Rfid authentication protocol for low-cost tags. In Gligor V. D., Hubaux J.-P., and Poovendran R., editors. *WISEC*. ACM, New York, NY, pages 140–147, 2008.
51. P. Traynor, R. Kumar, H. Choi, G. Cao, S. Zhu, and T. La Porta. Efficient hybrid security mechanisms for heterogeneous sensor networks. *IEEE Transactions on Mobile Computing*, 6(6):663–677, 2007.
52. R. Watro, D. Kong, S.-f. Cuti, C. Gardiner, C. Lynn, and P. Kruus. Tinypk: Securing sensor networks with public key technology. In *SASN '04: Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*. ACM, New York, NY, pages 59–64, 2004.
53. J. Wolkerstorfer. Invited talk: Scaling ecc hardware to a minimum., 2005. Presentation.
54. Q. Zhang, T. Yu, and P. Ning. A framework for identifying compromised nodes in wireless sensor networks. *ACM Transactions on Information and System Security*, 11(3):1–37, 2008.
55. W. Zhang, N. Subramanian, and G. Wang. *Lightweight and compromise resilient message authentication in Sensor Networks*, In: INFOCOM pages 1418–1426 (May 2008).
56. S. Zhu, S. Setia, and S. Jajodia. Leap: efficient security mechanisms for largescale distributed sensor networks. In: *CCS '03: Proceedings of the 10th ACM Conference on Computer and Communications Security*. ACM Press, New York, NY, pages 62–72, 2003.
57. S. Zhu, S. Setia, S. Jajodia, and P. Ning. Interleaved hop-by-hop authentication against false data injection attacks in sensor networks. *ACM Transaction on Sensor Networks*, 3(3):14, 2007.

Chapter 23
Key Management in Sensor Networks

Dahai Xu, Jeffrey Dwoskin, Jianwei Huang, Tian Lan, Ruby Lee, and Mung Chiang

Abstract Secure communications in wireless ad hoc networks require setting up end-to-end secret keys for communicating node pairs. It is widely believed that although being more complex, a probabilistic key predistribution scheme is much more resilient against node capture than a deterministic one in lightweight wireless ad hoc networks. Supported by the surprisingly large successful attack probabilities (SAPs) computed in this chapter, we show that the probabilistic approaches have only limited performance advantages over deterministic ones. We first consider a static network scenario as originally considered in the seminal paper by Eschenauer and Gligor [9], where any node capture happens after the establishment of all pairwise links. In this scenario, we show that the deterministic approach can achieve a performance as good as the probabilistic one. In a mobile network scenario, however, the probabilistic key management as described in [9] can lead to a SAP of one order of magnitude larger than the one in a static network due to node fabrication attacks.

The above analysis motivates us to propose two low-cost secure-architecture-based techniques to improve the security against such attacks. Our new architectures, specifically targeted at the sensor-node platform, protect long-term keys using a root of trust embedded in the hardware System-on-a-Chip (SoC). This prevents an adversary from extracting these protected long-term keys from a captured node to fabricate new nodes. The extensive simulation results show that the proposed architecture can significantly decrease the SAP and increase the security level of key management for mobile ad hoc networks.

Finally, we develop an analytical framework for the on-demand key establishment approach. We propose a novel security metric, the REM resilience vector, to quantify the resilience of any key establishment schemes against Revealing, Erasure, and Modification (REM) attacks. Our analysis shows that previous key establishment schemes are vulnerable under REM attacks. Relying on the new security metric, we prove a universal bound on achievable REM resilience vectors for any on-demand

D. Xu (✉)
AT&T Labs - Research, 180 Park Ave, Building 103, Florham Park, NJ 07932, USA
e-mail: dahaixu@research.att.com

The AT&T Labs - Research, The work was done when Xu was in Princeton University.

key establishment scheme. This bound that characterizes the optimal security performance analytically is shown to be tight, as we propose a REM-resilient key establishment scheme which achieves any vector within this bound. In addition, we develop a class of low-complexity key establishment schemes which achieve nearly optimal REM attack resilience.

23.1 Introduction

23.1.1 Motivation

Lightweight ad hoc networks typically consist of nodes that are distributed and have very limited computation and energy resources. Examples include portable mobile devices and tiny low-cost sensors used for environment surveillance and emergency response. Providing secure communication over this kind of network is challenging. Various key management schemes have been proposed with an attempt to provide a highly secure communication environment in lightweight ad hoc networks against various malicious attacks. Among the proposed schemes, *symmetric* key predistribution schemes (e.g., [1, 5, 15, 16, 23–25, 27]) are more suitable to the lightweight ad hoc network than *asymmetric* public-key schemes, because the former schemes require less resources (e.g., battery, memory, and computation power) and there is no need for a trusted third party for authorization.

There are two main approaches for symmetric key predistribution: *probabilistic* (e.g., [4, 5, 9, 17, 22, 27]) and *deterministic* (e.g., [2, 19, 30, 31]). In a probabilistic approach, the keys in each node's key ring are randomly chosen from a large key pool. In a deterministic approach, the key ring is chosen deterministically. In general, probabilistic approach end up with a large key pool, a larger key ring per node, and poorer network connectivity than a deterministic one.[1] On the other hand, a typical deterministic algorithm preloads each node with a single common key and reaches connectivity of 100%. More related references can be found in the survey [3].

It is often believed that a typical probabilistic scheme is much more resilient against node captures than a typical deterministic approach [4, 5, 9, 32], thus making probabilistic schemes popular despite their clear disadvantage on many other metrics. *In this chapter, we show that the probabilistic approaches have only limited advantages over deterministic approaches even when considering node capture*. Our performance measurement is the *Successful Attack Probability* (SAP). In particular, we consider an attack on a pairwise link between two authorized nodes to be successful if a compromised node can intercept and decipher the information transmitted through that link.

[1] For example, the probabilistic scheme in [9] requires preloading each node with 83 keys out of a key pool size of $10,000$ and achieves a local direct connectivity of 50%.

23.1.2 Summary of Our Study Between Representative Probabilistic and Deterministic Schemes

The probabilistic scheme was first proposed in the seminal and widely cited paper by Eschenauer and Gligor [9], and we call the corresponding scheme the *EG scheme*. It consists of three phases: *key distribution*, *shared key discovery*, and *path-key establishment*. In the key distribution phase, each node is loaded with k keys randomly chosen from a large key pool of size m, where $k \ll m$. The shared key discovery is the process of establishing a pairwise link between two neighbor nodes if they share one or more key(s). Finally, in the path-key establishment phase, a pairwise link is established between any two neighboring nodes who do not share any key but can establish a path between them through one or more relay nodes. In this case, a path-key is sent from one node to its neighbor through the relay(s), and then a link is established similarly to the shared-key discovery phase.

A typical deterministic scheme uses only a single common key, and each node is preloaded with the same initial key. After the deployment, each pair of neighbor nodes exchanges the messages encrypted by the common initial key to derive a unique (and are often random) key for all later communications between them.

Throughout the chapter, we will compare the performance of probabilistic and deterministic key management schemes based on the EG scheme [9] and single common key scheme. *We will show that the probabilistic scheme is not significantly better than the deterministic scheme measured in terms of SAP.* Since the single common key is one of the simplest deterministic schemes, any further improvement over it (e.g., [31]) will only reinforce our conclusion.

We consider two network scenarios: *static network* and *mobile network*. In a static network, all pairwise links have been established before an adversary captures any node. This is the case previously considered in [9]. This could happen, for example, if all nodes are deployed almost at the same time and remain stationary after deployment. In contrast, in a mobile network, an adversary can capture a node before all pairwise links have been established. This is true for a network where nodes are constantly on the move and need to establish new links. This includes, for example, a sensor network of buoys floating freely on the ocean to gather environmental data[28], or a network consisting of sensors moving around in an unknown environment to form reasonable coverage [13].

In a static network, the single common key deterministic scheme can achieve almost perfect resiliency against node capture (i.e., SAP $\approx$ 0). This is because the initial common key can be deleted permanently from all nodes after the establishment of all pairwise keys (as in [30]). Since all pairwise keys are randomly generated and known only to the corresponding two neighbor nodes, they cannot be deduced by a captured node even if the common initial key is disclosed. In the EG scheme, however, the SAP equals k/m with only one captured node where each neighbor node pair uses one of the shared keys to encrypt the communication. It is possible to reduce the SAP to almost 0 as in the single common key case if two neighbor nodes also generate a random key for future communication. In short, the deterministic scheme can achieve performance as good as the probabilistic approach in a static network, but with much lower complexity.

In a mobile network, the single common key deterministic scheme could lead to an SAP as high as 100% if the common initial key is obtained by an adversary before any link is established. However, we show that the EG algorithm is also quite vulnerable in this case and may lead to a value of SAP one order of magnitude larger than in the static network case (e.g., as high as 60%), especially when the adversary can fully utilize the keys obtained from several compromised nodes. The intuition for the surprising result is as follows. In the static network, there is only one way to attack a link successfully, i.e., knowing the key with which the communications on that link is encrypted. In a mobile network, however, a compromised node can also attack a link by acting as a relay during the path-key establishment phase. By intercepting the key information that is being relayed, a compromised node can figure out the key which the two authorized nodes will use for future mutual communication. This new *man-in-the-middle* attack opportunity can significantly increase the value of SAP for a probabilistic approach, since nodes frequently use a relay for link establishment.

After thus re-examining the performance difference between probabilistic and deterministic key predistribution schemes, we propose two secure hardware-based techniques, specifically targeted to the sensor-node platform, which protect long-term keys. Such techniques can be used to improve the performance of both deterministic and probabilistic key management schemes for mobile networks. They ensure that protected secrets cannot be extracted from a captured node. This is the first step toward building a comprehensive low-cost secure-hardware design for sensor nodes.

The rest of this chapter is organized as follows. In Section 23.2, we calculate the values of SAP in both static and mobile networks, with a focus on the probabilistic approach (i.e., EG scheme). In Sect. 23.3, we propose processor architecture-based techniques for securing secret keys and critical software on a node. In Sect. 23.4 we analyze the security of the proposed architecture under several specific attacks. In Sect. 23.5, we validate the analytical results from Sect. 23.2 and the security improvement of the proposed architecture with simulations based on a C++ simulator. In Sect. 23.6, we reexamine other probabilistic key predistribution schemes. In Sect. 23.7, we introduce our REM resilient key establishment framework and a low-complexity protocol. We conclude in Sect. 23.8.

23.2 Fragility Analysis for Probabilistic Key Management

In this section, we first review the results in [9], where the successful attack probability (SAP) is calculated for a static network. We then consider a mobile network and show how the value of SAP is significantly larger in that case. We only consider the attacks on the *pairwise* link between two authorized nodes that are within each other's communication range. The SAP will be even higher if A and B are far away and can only be connected with a multi-hop path, since a successful attack on any hop will jeopardize the confidentiality of the whole communication.

The establishment of a link requires two neighbor nodes, A and B, to be able to encrypt the communication over such a link using a common key. This could be achieved in two ways:

(i) A and B share a key within their preloaded key rings, thus can establish the link directly.

(ii) A and B do not share a key initially and need to exchange additional information through one or more relay nodes, with whom the pairwise links have already been established. For example, A can randomly choose an unused key from its key ring and send it to B through the relay node(s). Then A and B can use this key to encrypt the pairwise key between them.

In either case, SAP of the link between A and B is defined as

$$SAP \triangleq P(A \otimes B | A \leftrightarrow B)$$

where $A \otimes B$ denotes the event that the link between A and B is successfully attacked, and $A \leftrightarrow B$ denotes the event that A and B establish a link between them. Since a link can only be attacked if it has been established, we have (23.1) and (23.2):

$$P(A \otimes B \cap A \leftrightarrow B) = P(A \otimes B) \tag{23.1}$$

$$SAP = \frac{P(A \otimes B)}{P(A \leftrightarrow B)} \tag{23.2}$$

All the notation used in this section is defined in Table 23.1 to enable a cleaner presentation of later derivations. A, B, and C denote three generic nodes, and $\mathbb{C}^h$ denotes a set of h nodes. Each node is preloaded with a key ring of k randomly chosen keys out of a key pool of size m.

Table 23.1 Summary of notation

Notations	Meaning
$A \leftrightarrow B$	A and B establish a pairwise link between them
$A \leftrightarrow \mathbb{C}^h \leftrightarrow B$	A and B communicate through one node in $\mathbb{C}^h$
$A \otimes B$	The link between A and B is successfully attacked
$A \sharp B$	A and B share at least one key
$(A \sharp B) \lhd C$	C has all the keys (≥ 1) shared by A and B
$(A \sharp B) \lhd \mathbb{C}^h$	At least one node of $\mathbb{C}^h$ has all the keys (≥ 1) shared by A and B
$(A, B) \sharp \mathbb{C}^h$	At least one node in $\mathbb{C}^h$ shares at least one key with A and at least one key with B
$(A, B) \sharp \mathbb{C}^h_r$	Exactly r nodes out of $\mathbb{C}^h$, each of which shares at least one key with A and at least one key with B

23.2.1 SAP for a Static Network

If a compromised node wants to attack an established link, it needs to know the key that is used to encrypt the link. Therefore, a compromised node can successfully attack an existing link with probability k/m, as stated in [9].

23.2.2 SAP for a Mobile Network

In a mobile network, a compromised node C can attack the link between A and B in three ways:

(i) If A and B share a key initially and establish the link directly, then C needs to know the key chosen by A and B to encrypt the link.
(ii) If A and B do not share a key initially and use C as a relay, then C can get the desired information while relaying the information between A and B. A first communicates with C via encrypted messages protected by shared key K_{ac}. C decrypts this with K_{ac} giving it access to the plain text message, re-encrypts it with K_{cb}, a key it shares with node B, and then sends the re-encrypted message to B. This sets C up as a man-in-the-middle eavesdropper between A and B, since C can see the plain text of all messages going from A to B.
(iii) If A and B do not share a key and do not choose C within the relay path, C can still attack the communication between A and B by either eavesdropping on the links along the relay path or attacking the eventual pairwise link established between A and B, if it has any of the keys used for these links.

Overall, the value of SAP depends on the number of compromised nodes and authorized nodes within both A and B's communication range, as well as how A and B choose the relay nodes. To simplify the analysis, we only consider cases (i) and (ii), and further assume only one node relay in case (ii). In the simulation in Sect. 23.5, we calculate SAP for all three cases.

It will be useful to know the probability of sharing at least one key between any two nodes in the network. Denote δ_m^k as the probability that any two nodes A and B do *not* share any key, then

$$\delta_m^k \triangleq P\left(\overline{A \sharp B}\right) = \binom{m-k}{k} \Big/ \binom{m}{k}$$

where $A \sharp B$ denotes A and B share at least one key. The value of δ_m^k can be either accurately calculated as $\prod_{i=0}^{k-1}(m-k-i)/(m-i)$, or approximated using Stirling's approximation for $n!$ as in [9], i.e.,

$$\delta_m^k = \frac{\binom{m-k}{k}}{\binom{m}{k}} \approx \frac{\left(1-\frac{k}{m}\right)^{2(m-k+0.5)}}{\left(1-\frac{2k}{m}\right)^{m-2k+0.5}}$$

Then the probability of A and B sharing at least one key is

$$P(A\sharp B) = 1 - \delta_m^k \tag{23.3}$$

For example, if $k = 83,\ m = 10000,\ P(A\sharp B) \approx 50\%$.

Next, we derive the value of SAP based on the number of authorized users and compromised users within both A and B's communication range. We start with the simplest case, where there is only one compromised node available. We then consider the case where there are h compromised nodes. Finally, we consider the case with h compromised nodes and g authorized nodes.

23.2.2.1 Scenario I: Only One Compromised Node C is Within Both A and B's Communication Range

Depending on whether A and B share a key initially, they may establish the pairwise link with or without the relay of C. The probability of successfully establishing the link is (23.4) and the probability of attacking the link is (23.5).

$$P(A \leftrightarrow B) = P(A\sharp B) + P((A\sharp C \cap B\sharp C) \cap \overline{A\sharp B}) \tag{23.4}$$

$$P(A \otimes B) \geq P((A\sharp B) \lhd C) + P((A\sharp C \cap B\sharp C) \cap \overline{A\sharp B}) \tag{23.5}$$

Here $((A\sharp B) \lhd C)$ means that A and B share at least one key, and all the shared keys between A and B are within the key ring of node C. Since we ignore the case where C only knows a subset of the shared keys between A and B, where C still has a chance to successfully attack the link between A and B, we have an inequality in (23.5) instead of an equality.

Let us calculate each term in (23.4) and (23.5). We know the value of $P(A\sharp B)$ from (23.3). Also,

$$\begin{aligned}
&P(A\sharp C \cap B\sharp C|\overline{A\sharp B}) \\
&= 1 - P(\overline{A\sharp C}) - P(\overline{B\sharp C}) + P(\overline{A\sharp C} \cap \overline{B\sharp C}|\overline{A\sharp B}) \\
&= 1 - 2\delta_m^k + \binom{m-2k}{k}\Big/\binom{m}{k} \\
&= 1 - 2\delta_m^k + \binom{m-k}{k}\Big/\binom{m}{k} \cdot \binom{m-2k}{k}\Big/\binom{m-k}{k} \\
&= 1 - 2\delta_m^k + \delta_m^k \cdot \delta_{m-k}^k
\end{aligned}$$

Define

$$\phi_m^k \triangleq P(A\sharp C \cap B\sharp C | \overline{A\sharp B})$$

we then have

$$P(A\sharp C \cap B\sharp C \cap \overline{A\sharp B}) = P(\overline{A\sharp B}) \cdot P(A\sharp C \cap B\sharp C|\overline{A\sharp B}) = \delta_m^k \phi_m^k \tag{23.6}$$

Thus from (23.3), (23.4), and (23.6),

$$P(A \leftrightarrow B) = 1 - \delta_m^k + \delta_m^k \phi_m^k$$

Meanwhile,

$$\begin{aligned} &P((A\sharp B) \lhd C) \\ &= \sum_{i=1}^{k} \left(\binom{k}{i} \cdot \left(\frac{\binom{m-k}{k-i}}{\binom{m}{k}} \right) \cdot \left(\frac{\binom{m-i}{k-i}}{\binom{m}{k}} \right) \right) \\ &\geq \binom{k}{1} \cdot \left(\frac{\binom{m-k}{k-1}}{\binom{m}{k}} \right) \cdot \left(\frac{\binom{m-1}{k-1}}{\binom{m}{k}} \right) \\ &= k \left(\frac{k}{m-2k+1} \cdot \frac{\binom{m-k}{k}}{\binom{m}{k}} \right) \cdot \left(\frac{\frac{(m-1)!}{(k-1)!(m-k)!}}{\frac{m!}{k!(m-k)!}} \right) \\ &= \frac{\delta_m^k k^3}{m(m-2k+1)} \end{aligned} \tag{23.7}$$

whereas in (23.7), for simplicity we ignore the event that A, B, and C share more than one key. Define

$$\gamma_m^k \triangleq P((A\sharp B) \lhd C)$$

we then have

$$SAP = \frac{P(A \otimes B)}{P(A \leftrightarrow B)} \geq \frac{\gamma_m^k + \delta_m^k \phi_m^k}{1 - \delta_m^k + \delta_m^k \phi_m^k}$$

23.2.2.2 Scenario II: *h* Compromised Nodes are Within Both A and B's Communication Range

We use $\mathbb{C}^h$ to denote the set of h compromised nodes. Since

$$\begin{aligned}&P((A,B)\sharp\mathbb{C}^h\cap\overline{A\sharp B})\\ =&P(\overline{A\sharp B})\cdot P((A,B)\sharp\mathbb{C}^h|\overline{A\sharp B})\\ =&P(\overline{A\sharp B})\cdot(1-(1-P(A\sharp C\cap B\sharp C|\overline{A\sharp B}))^h)\\ =&\delta_m^k\cdot\left(1-\left(1-\phi_m^k\right)^h\right)\end{aligned}$$

then using a similar argument as in Scenario I, we have

$$\begin{aligned}SAP&\geq\frac{P((A\sharp B)\lhd\mathbb{C}^h)+P((A,B)\sharp\mathbb{C}^h\cap\overline{A\sharp B})}{P(A\sharp B)+P((A,B)\sharp\mathbb{C}^h\cap\overline{A\sharp B})}\\ &\geq\frac{1-\left(1-\gamma_m^k\right)^h+\delta_m^k\cdot\left(1-\left(1-\phi_m^k\right)^h\right)}{1-\delta_m^k+\delta_m^k\cdot\left(1-\left(1-\phi_m^k\right)^h\right)}\end{aligned}$$

23.2.2.3 Scenario III: *h* Compromised Nodes and *g* Authorized Nodes are Within Both A and B's Communication Range

In this case, if A and B do not share any key initially and need to communicate through a relay, a successful attack can happen if one compromised node is chosen as the relay. Assuming there are a total of *a qualified* relays (i.e., nodes who can establish pairwise links with both A and B), b out of which are compromised nodes. Denote μ_a^b as the probability of A and B picking a compromised node as the relay, which can have different values depending on the specific attack models (details in the next section).

The probability of having r useable relays out of all h compromised nodes when A and B do not share keys is

$$\begin{aligned}P\left((A,B)\sharp\mathbb{C}_r^h|\overline{A\sharp B}\right)&=\binom{h}{r}\left(P(A\sharp C\cap B\sharp C|\overline{A\sharp B})\right)^r\left(1-P(A\sharp C\cap B\sharp C|\overline{A\sharp B})\right)^{h-r}\\ &=\binom{h}{r}\left(\phi_m^k\right)^r\left(1-\phi_m^k\right)^{h-r}\end{aligned}$$

Similarly, the probability of having w useable relays out of all g authorized nodes when A and B do not share keys is

$$P\left((A,B)\sharp\mathbb{C}_w^g|\overline{A\sharp B}\right)=\binom{g}{w}\left(\phi_m^k\right)^w\left(1-\phi_m^k\right)^{g-w}\tag{23.8}$$

Then the probability of sending a message through a compromised node given the existence of h compromised nodes, g authorized nodes, and A and B do not share any key is

$$
\begin{aligned}
&P(A \leftrightarrow \mathbb{C}^h \leftrightarrow B|\overline{A\sharp B}) \\
&= \sum_{r=1}^{h}\sum_{w=0}^{g} \mu_{r+w}^{r} \left(P\left((A,B)\sharp\mathbb{C}_r^h|\overline{A\sharp B}\right)\cdot P\left((A,B)\sharp\mathbb{C}_w^g|\overline{A\sharp B}\right)\right) \\
&= \sum_{r=1}^{h}\sum_{w=0}^{g} \mu_{r+w}^{r} \left(\binom{h}{r}\binom{g}{w}\left(\left(\phi_m^k\right)^{r+w}\left(1-\phi_m^k\right)^{h+g-(r+w)}\right)\right)
\end{aligned}
$$

Since

$$P(A \leftrightarrow \mathbb{C}^h \leftrightarrow B \cap \overline{A\sharp B}) = P(A\sharp B)\cdot P(A \leftrightarrow \mathbb{C}^h \leftrightarrow B|\overline{A\sharp B})$$

we have the following lower bound on SAP:

$$
\begin{aligned}
SAP &= \frac{P(A \otimes B)}{P(A \leftrightarrow B)} \\
&\geq \frac{P((A\sharp B) \lhd \mathbb{C}^h) + P(A \leftrightarrow \mathbb{C}^h \leftrightarrow B \cap \overline{A\sharp B})}{P(A\sharp B) + P(A \leftrightarrow \mathbb{C}^{h+g} \leftrightarrow B \cap \overline{A\sharp B})} \\
&= \frac{1-\left(1-\gamma_m^k\right)^h + \delta_m^k\cdot\left(\sum_{r=1}^{h}\sum_{w=0}^{g}\mu_{r+w}^{r}\left(\binom{h}{r}\binom{g}{w}\left(\left(\phi_m^k\right)^{r+w}\left(1-\phi_m^k\right)^{h+g-(r+w)}\right)\right)\right)}{1-\delta_m^k+\delta_m^k\cdot\left(1-\left(1-\phi_m^k\right)^{h+g}\right)}
\end{aligned}
$$

23.2.2.4 Numerical Results

Table 23.2 shows the SAP for different values of h and g based on the previous analysis. The key ring size is $k = 83$, with a key pool size of $m = 10,000$.

Table 23.2 Successful attack probability (SAP) for different numbers of authorized nodes (g) and compromised nodes (h). We assume there are a total of a qualified relays, b out of which are compromised nodes. μ_a^b is the probability of picking a compromised node as the relay. The key pool size $m = 10000$, the preloaded key ring size $k = 83$, and the original SAP estimation is hk/m

h	$g = 0$	$g = 10$		$g = 20$		hk/m
		$\mu_a^b = b/a$	$\mu_a^b = 1$	$\mu_a^b = b/a$	$\mu_a^b = 1$	
1	20.4%	4.7%	13.0%	2.7%	12.8%	0.8%
2	31.1%	8.8%	22.7%	5.1%	22.4%	1.7%
3	37.6%	12.3%	30.0%	7.4%	29.7%	2.5%
4	41.9%	15.3%	35.5%	9.5%	35.2%	3.3%
5	44.8%	18.0%	39.7%	11.4%	39.5%	4.2%
6	46.9%	20.4%	42.9%	13.2%	42.7%	5.0%
7	48.5%	22.5%	45.4%	15.0%	45.2%	5.8%
8	49.7%	24.4%	47.3%	16.6%	47.2%	6.6%
9	50.6%	26.2%	48.8%	18.1%	48.7%	7.5%

Several observations are in order. When the probability of picking a compromised node as the relay $\mu_a^b = b/a$, the SAP increases with h (the number of compromised nodes) under a fixed g (the number of authorized nodes). When $\mu_a^b = 1$, the general trend is similar, but the SAP is not very sensitive to g between the cases of $g = 10$ and $g = 20$, since A and B will always choose a compromised node as relay if possible. Comparing with the value of SAP estimated in [9], which is approximated as hk/m, the SAP in Table 23.2 is much larger. For example, with $\mu_a^b = b/a$, $h = 9$ and $g = 20$, we have an SAP of 18.1%, as opposed to $hk/m = 7.5\%$. The value of SAP further increases when $\mu_a^b = 1$.

The value of μ_a^b depends heavily on the attack model used by the compromised nodes. We define two attack models, *honest attack* and *smart attack*. In an honest attack, the relay nodes are randomly chosen and $\mu_a^b = b/a$. In a smart attack, however, the compromised nodes will improve the value of μ_a^b by various methods. In a *smart attack with incentive*, the compromised nodes provide incentives for nodes A and B to choose one of them as a relay. If the choice of relay is determined by a shortest path routing protocol, the compromised nodes can announce distance metrics of the links connected to them smaller than the actual values. If the choice of relay is based on energy efficiency, the compromised nodes can pretend to be very energy efficient. In most cases, the incentives provided by the compromised nodes can make the value of μ_a^b very close to 1. In *a smart attack with virtual node fabrication*, each compromised node is able to collect the keys from all other compromised nodes and can then fabricate up to $\binom{hk}{k}$ nodes with distinct key rings. The number will be very large if $h \geq 2$. For example, when two nodes are captured with non-overlapping key rings, then

$$\binom{2k}{k} = \frac{(2k)!}{k!} \approx \frac{\sqrt{2\pi}(2k)^{2k+0.5}e^{-2k}}{\left(\sqrt{2\pi}(k)^{k+0.5}e^{-k}\right)^2} = \frac{2^{2k+0.5}}{\sqrt{2\pi k}} \tag{23.9}$$

which is around 5.8×10^{48} if $k = 83$. As a result, the value of μ_a^b will be closer to 1 as the number of fabricated nodes increases.

23.3 Secret-Protecting Processor Architecture

The analysis in the previous section is based on the assumption that an adversary can obtain the long-term key information from the captured nodes. In particular, we assume an adversary with physical access to the device, so software protections are easily bypassed. The keys are accessible to an adversary if the existing software is exploited or if the software is replaced entirely with malicious code. He might also read the keys directly from a flash memory chip or other permanent storage when the device is offline.

We propose a solution that protects secrets by storing them inside the System on a Chip (SoC) [8]. The chip includes the processor core and main memory. It is quite expensive for an adversary to remove the packaging and directly probe the registers and memory. The SoC chip can further implement physical tamper-resistance mechanisms, which will clear the secrets whenever probing attempts are detected, cut the power supply to the chip, and erase any intermediate data based on those secrets. Therefore, the assumption of protected on-chip secrets is valid for a large class of attacks.

Our solution is to provide a Secret Protecting (SP) architecture which minimizes the trusted computing base (TCB) of hardware and software that has to be fully correct, verified, and trusted in order to protect the long-term secrets. Our TCB comprises some SP hardware features (described below) and a small Trusted Software Module (TSM) that performs the key management on the node.

We first present the Reduced Sensor-mode SP that is suitable for the simplest sensors. We then extend the solution for slightly more capable sensors. Our work is inspired by the SP architecture proposed for general-purpose microprocessors [7, 20], but stripped to the bare minimum for sensors with very constrained computing and storage resources.

23.3.1 Reduced Hardware Architecture

The simplest version of our architecture, Reduced Sensor-mode SP, is shown in Fig. 23.1. It only requires one new register — the Device Key—and one bit to indicate protected mode. A Trusted Software Module (TSM) is stored in the on-chip instruction EEPROM. The long-term keys for the probabilistic key management scheme, provided by a central authority, are stored in the on-chip data EEPROM. A portion of the main memory of the node is reserved for the TSM Scratchpad Memory.

The key feature of our design is that the TSM is the only software module that can use the Device Key and the protected long-term keys. Since the TSM code is stored within the trusted SoC chip in ROM, it cannot be changed by other software—whether by a malevolent application or by a compromised operating system. Similarly, the long-term keys never leave the SoC chip. Any intermediate data (which may leak key bits) generated during TSM execution are placed in the TSM scratchpad memory and also never leave the SoC chip. We will discuss how this prevents node fabrication attacks in Sect. 23.4.

The TSM code is stored on-chip in a segment of the existing instruction EEPROM along with other system software for the node. Correspondingly, the long-term keys from the authority are stored in a TSM segment of the data EEPROM. The keys are encrypted with the device key or with another encryption key derived from it by the TSM.

The device key is the SP master key and is protected by the processor hardware; it can only be used by the TSM running in protected mode and can never be read by any other software.

When the unprotected software wants to make use of protected keys, it calls the TSM. The TSM functions access the protected keys, perform the requested

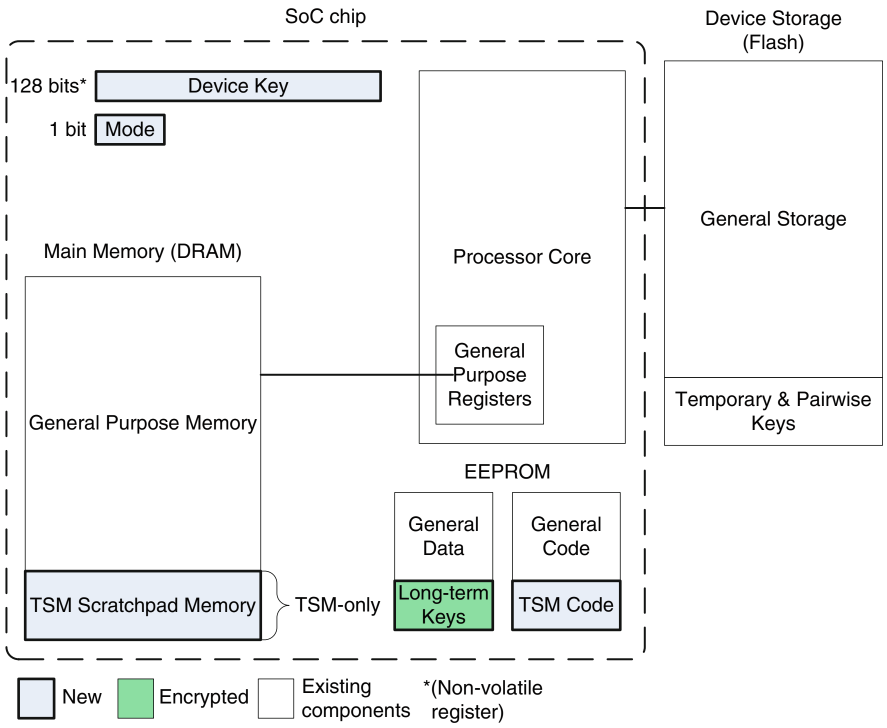


Fig. 23.1 Reduced sensor-mode SP

operation, and return the results, never revealing the protected keys themselves to the unprotected software. Each TSM function starts with a *Begin_TSM* instruction, which disables interrupts, sets the protected mode bit, and enters protected mode for the next instruction. *Begin_TSM* is only valid for code executing from the instruction-EEPROM; any code executed from main memory or off-chip storage cannot enter the protected mode at all. The end of the TSM code is indicated by the *End_TSM* instruction which clears the mode bit and re-enables interrupts. Table 23.3 shows the set of instructions used only by the TSM and for initialization in the Sensor-mode SP architectures.

The TSM Scratchpad Memory is a section of main memory reserved for the exclusive use of the TSM. It is addressed separately from the regular on-chip memory and accessed only with special *Secure_Load* and *Secure_Store* instructions (see Table 23.3). These new instructions are available only to the TSM, making it safe for storing sensitive intermediate data in the TSM scratchpad memory. The TSM can also use this extra space to spill general registers, to decrypt and store keys, and to encrypt data for storage in regular unprotected memory.

Initialization of a new device takes place at the authority's depot. First the authority must generate a new random device key. Long-term keys and other secrets are encrypted with it are then stored along with the TSM code on the on-chip EEPROM.

Table 23.3 New sensor-mode SP instructions

Instruction	Description
Begin_TSM	Begins execution of the TSM
End_TSM	Ends execution of the TSM
Secure_Store	Secure store from processor to TSM scratchpad memory (TSM only)
Secure_Load	Secure load from TSM scratchpad memory to processor (TSM only)
DeviceKey_Read	Read the Device Key (TSM only)
DeviceKey_Set	Sets the Device Key register. First clears the TSM scratchpad memory
ASH_Set	Sets the ASH register. First clears the device key and TSM scratchpad memory

Next it uses the *DeviceKey_Set* instruction to store the device key. Finally, any other unprotected software and data can be copied to the flash storage.

Any time the Device Key register is set (or cleared), the processor will automatically clear the TSM scratchpad memory, wiping any intermediate data that are protected by the old key. If in protected mode at the time, the mode bit is also cleared along with the general-purpose registers. Similarly, the processor will clear the device key upon writing to either the instruction or data EEPROM; this in turn clears the other intermediate data.

23.3.2 Expanded Sensor-Mode SP Architecture

The Reduced Sensor-mode SP architecture is ideal for the smallest sensor nodes which use minimal software and have very limited resources. In slightly larger lightweight sensor nodes, the software will be more complex. The additional applications that run on this sensor combined with the TSM and long-term keys will be too large to store on-chip. This greater flexibility in the sensor also requires additional support for security. Hence, we propose the Expanded Sensor-mode SP architecture shown in Fig. 23.2.

The TSM code and encrypted long-term keys are moved to the off-chip device storage. This makes them susceptible to modification by other software or through physical attacks. Therefore we must verify their integrity before they can be used. To do this, we add a new register—the Authority Storage Hash (ASH), a hardware hashing engine (implementing SHA-1, MD5, or another cryptographic hash function), a small ROM, and an additional initialization instruction.

The ASH register contains a hash over the entire memory region of the TSM code and long-term keys. It is set by the authority during initialization and is rechecked by the processor each time the TSM is called. The checking code is stored in the on-chip ROM and is fixed and therefore safe from modification; it uses the hardware hashing engine to compute the hash over the TSM code and the encrypted keys. When *Begin_TSM* is called, the processor disables interrupts and jumps to the TSM-checking routine. If the hash check succeeds, the protected mode bit is set, and execution jumps to the newly verified TSM code. If the check fails, an exception is triggered. The *ASH_Set* instruction sets the ASH register and also causes the device

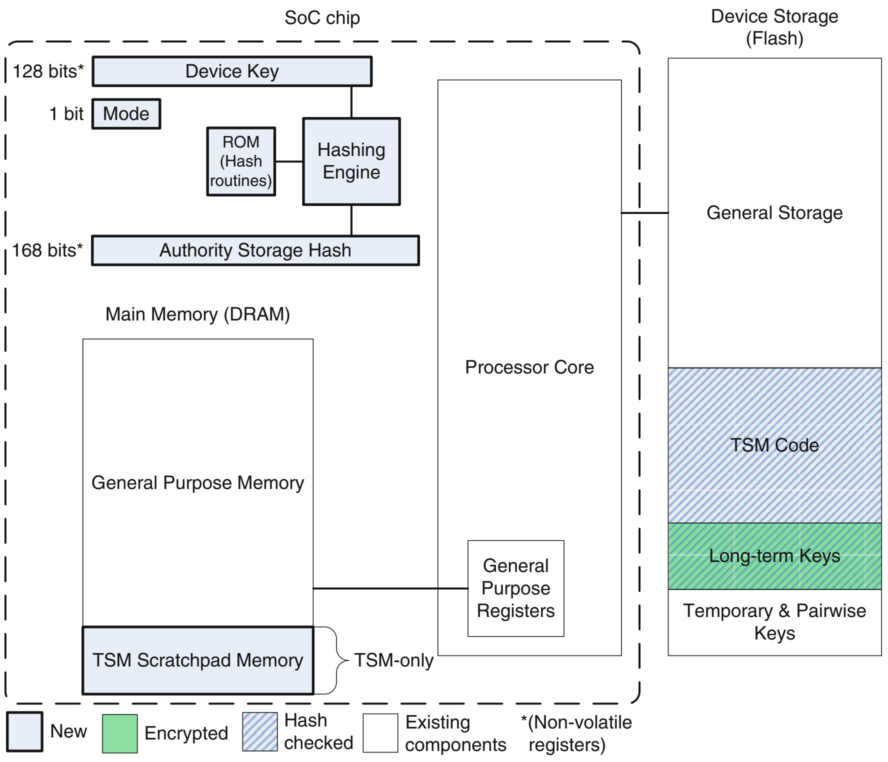


Fig. 23.2 Expanded sensor-mode SP

key register to be cleared. As the ASH is used to verify the TSM code, installing a new TSM requires writing a new value to the ASH value. Clearing the device key therefore ensures that a new TSM cannot be installed and still have access to the protected keys that belonged to another TSM.

23.4 Security and Economics Analysis of SP Architecture-Based Solution

23.4.1 Attacks on Protected Keys

Our new Sensor-mode SP architectures safeguard a sensor node's long-term keys, preventing extraction by an adversary in the event of node capture. The keys are always stored in encrypted form in the permanent storage in either on-chip EEP-ROM or off-chip storage. The adversary cannot obtain the device key needed to decrypt them. The device key never leaves the SP processor or its protected software environment. Therefore, rather than access the keys directly, regular software must call TSM functions which perform operations with the keys on its behalf. Thus, software can use the keys in any way permitted by the TSM, but can never extract the keys themselves even under physical attacks.

23.4.1.1 Node Fabrication Attacks

Without the SP protection, an adversary maximizes his SAP by cloning multiple copies of compromised nodes and combining their long-term keys. This increases his ability to observe link establishment and the likelihood of being used as a relay. With the SP protection, he cannot create any clones and is limited to using only the keys originally stored on the captured node.

23.4.1.2 Node Capture Attacks

Node capture attacks use long-term keys in the node to observe pairwise links between other nodes in the network. With SP, an adversary can no longer extract the keys. However, he can still change unprotected software which calls the TSM. A simple TSM might provide functions like *Encrypt(key, data*) and *Decrypt(key, data)*. The adversary can use the keys through this TSM interface to observe or attack pairwise links without ever seeing the actual keys. While we do not prevent node capture attacks outright, such attacks are limited since the adversary can only observe links within the communication range of the compromised node. We show in Sect. 23.5 that this severely limits the SAP, which is constrained by the number of captured nodes.

23.4.2 Attacks on Changing the TSM or the Device Key

The security of the long-term keys relies on the correctness and proper design of the authority's TSM. As part of the trusted computing base of the system, this software must not leak secrets it has access to. This includes any intermediate data written to general-purpose memory, placed in off-chip storage, or left in general registers when it exits. The TSM runs with interrupts disabled, so no other software will have an opportunity to observe its registers or modify its code or data while it is executing. If the TSM ever exists abnormally due to an exception, the processor clears the general registers before ending protected mode. Any other sensitive data will be in the TSM scratchpad memory which other software cannot access.

In order to circumvent the access control provided by the authority's TSM, the attacker might try to replace it with his own TSM or modify the existing TSM. In the Reduced Sensor-mode, the TSM and long-term keys are stored in on-chip EEPROM where they cannot be modified without also clearing the device key. In the Expanded Sensor-mode, the attacker could modify or replace the TSM code in off-chip storage. The hash checking routine will detect any such modifications made to the TSM before execution. We assume that the data in off-chip storage cannot be modified through a physical attack during execution. If this is not the case, the TSM and keys should first be copied to the general-purpose memory on-chip before being verified, where they will be safe from physical attacks.

Finally, if the attacker tries to modify the ASH register to match the new TSM code, the device key will be cleared, irrevocably cutting off his access to all of the keys that were encrypted with that device key. Clearing or setting the device key also clears the TSM scratchpad memory, so any intermediate data stored there that might have leaked secrets are also unavailable to the new TSM.

23.4.3 Economics Analysis

When considering low-cost sensors, any new hardware must be designed for high volume in order to keep down fabrication costs. Our Sensor-mode SP provides basic security primitives and a hardware root of trust using a design that is easily integrated into the SoC of standard embedded processors. It therefore supports a wide range of software protection mechanisms with only a slight increase in chip area.

Our hardware also provides physical security. SP prevents attacks by an adversary with physical control over a captured node, who tries to modify the code or data in storage while the device is in operation or offline. The physical integrity of the SoC itself is sufficient to prevent adversaries from probing the SP registers inside the chip, without requiring more costly tamper-proofing mechanisms in most cases.

23.5 Simulation Results

To verify our probability computations in Sect. 23.2 and demonstrate the improvement of security performance of the proposed architecture in Sect. 23.3, we evaluate the SAP of the probabilistic and deterministic key predistribution scheme (the EG scheme) through a simulator written in C++.

23.5.1 Comparison of Probabilistic and Deterministic Key Predistribution

To compare probabilistic and deterministic key predistribution schemes, we consider a unit disk network model, as shown in Fig. 23.3. A total of g authorized nodes (denoted by symbol A) are uniformly distributed in the unit disk. All the compromised nodes (including any virtually fabricated nodes) are placed at the center of the unit disk and denoted with symbol C. All nodes are assumed to have the same transmission range equal to the radius of the disk. This means an adversary can eavesdrop on any communication in the unit disk through the compromised nodes as long as it has the right key(s). Two neighbor nodes will set up a pairwise link directly if they share one or more keys. Otherwise, they will try to find a relay path through one or more nodes to exchange additional key information, so that

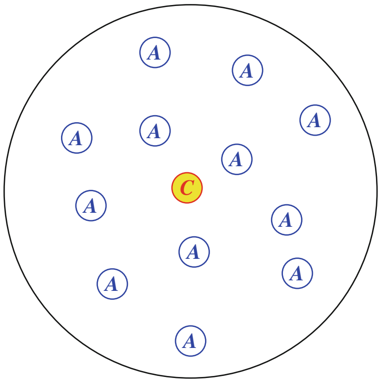


Fig. 23.3 Unit disk network model with a unit radius. The authorized nodes are uniformly distributed in the unit disk and denoted by the symbol A. The compromised and virtually fabricated nodes are placed in the center of the network and denoted by the symbol C. All nodes have the same communication range equal to the disk radius

they can set up pairwise link between them. When there is more than one qualified relay node available, the authorized nodes will choose a relay randomly in the case of $\mu_a^b = b/a$ (i.e., honest attack or finite virtual node fabrication), or search for a shortest relay path in an attack with incentive.[2] Any two nodes that are not neighbors cannot establish pairwise links among themselves. The main reason of using the above unit disk network model is to derive a uniform and fair metric (i.e., SAP) among various approaches where failing to attack is only due to the lacking of appropriate keys rather than the limitation of transmission range.

The SAP is calculated as the fraction of the links (among all the pairwise links) that can be eavesdropped by the compromised nodes. As we explained in Sect. 23.1, a basic deterministic scheme like single common key either enables nearly zero SAP in a static network, or leads to 100% SAP for the unit disk model in a mobile network. Hence, our focus here is to determine the SAP for the probabilistic key predistribution scheme (i.e., the EG scheme). All the simulation results are averaged over 10 sets of random seeds which affect the distribution of the authorized nodes within the unit disk, the key ring preloaded to each node and the choices in case of multiple qualified relays.

Figures 23.5, 23.6, and 23.7 illustrate the values of SAP under different assumptions on the number of compromised nodes (h), number of authorized nodes (g), and different attack models (honest attack, smart attack with incentive, or smart attack with fabrication). Unless otherwise specified, each node is preloaded with a key ring consisting of $k = 83$ keys that are randomly chosen from a key pool of size $m = 10,000$.

Fig. 23.5 shows the SAP for various values of h and g under the *honest attack*. For a fixed value of h, the SAP decreases when the density of authorized nodes increases. This is because in a denser network, there are more qualified relay nodes available between any two neighbor nodes; thus the probability of choosing a compromised node as the relay is smaller under honest attack. For a fixed number of

[2] In the simulation, the smart attack with incentive is approximated as setting the cost of the links adjacent to the compromised nodes as 0.9999 instead of as 1 unit (hop) for other authorized nodes.

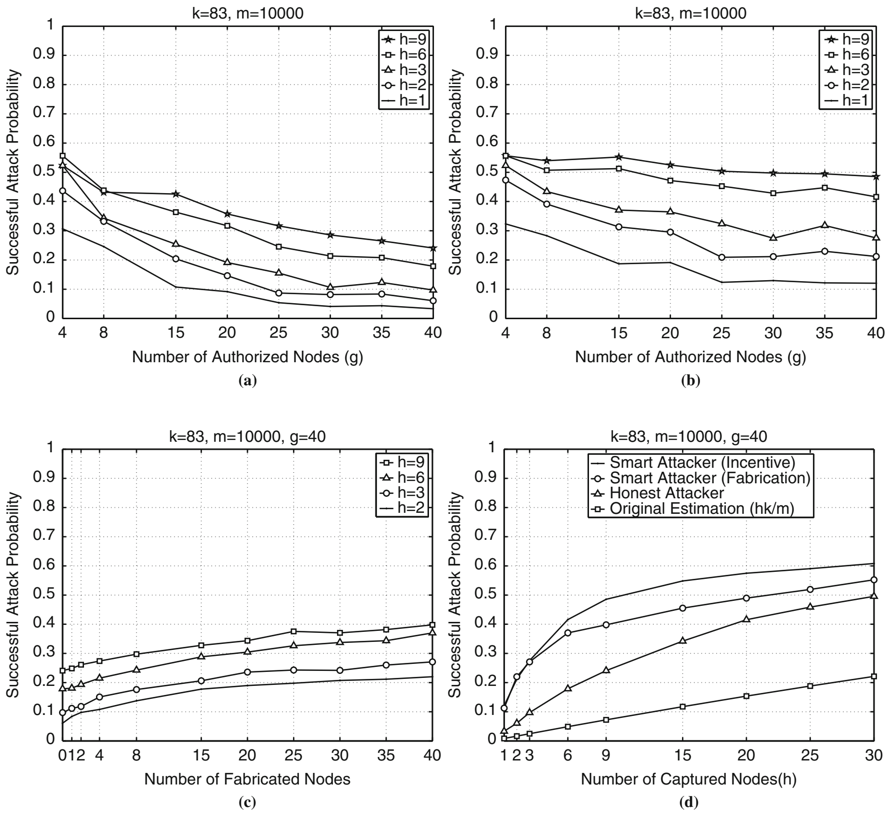


Fig. 23.4 Successful attack probability with various numbers of captured nodes (h) and authorized nodes (g), (**a**) Honest attack; (**b**) Smart attack; (**c**) Smart attack (node fabrication); and (**d**) Different attack models

authorized nodes g, a higher value of h increases the probability of picking a compromised node as the relay, thus leads to a higher value of SAP. In a network with 9 compromised nodes and 15 authorized nodes, the SAP could be as high as 42%.

Figure 23.4(b) shows the SAP for various values of h and g under the *smart attack with incentive*. In this case, two neighbor nodes without a common key will have a high chance to pick a compromised node as relay if it is qualified. There is a high probability of finding a qualified relay node among the compromised nodes when h is large, in which case the SAP is insensitive to the number of authorized nodes g. Similarly as in Fig. 23.5, a higher value of h also leads to a higher value of SAP. In a network with 40 authorized nodes and 9 compromised nodes, the SAP would be around 50%.

Figure 23.6 shows the SAP for the smart attack of various numbers of compromised nodes and different total numbers of virtually fabricated nodes. The total number of authorized nodes is kept at 40. The node fabrication is achieved as follows. All the keys collected from the h compromised nodes will constitute a

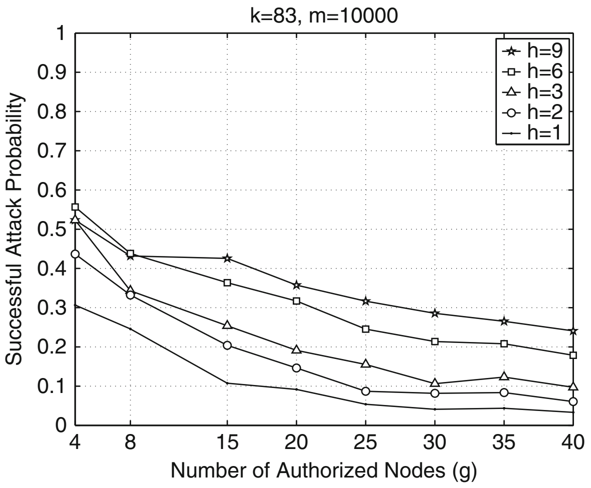


Fig. 23.5 Successful attack probability for various numbers of captured nodes (**h**) and authorized nodes (**g**) under an honest attack

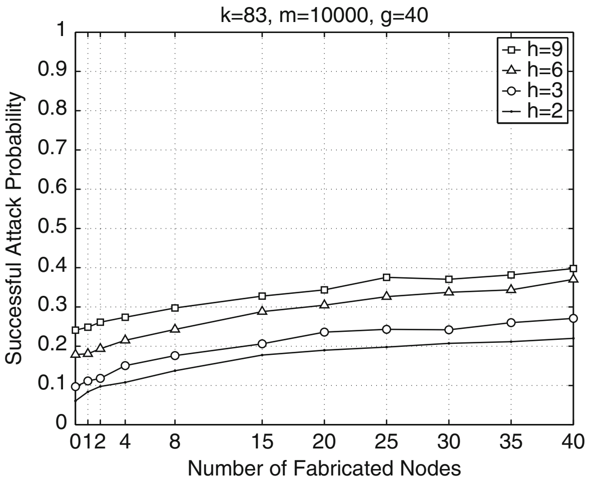


Fig. 23.6 Successful attack probability for various numbers of captured nodes (**h**) and different numbers of node fabrications in smart attack

compromised key pool. Then each fabricated node will be loaded with $k = 83$ keys randomly chosen from the compromised key pool. A larger number of fabricated nodes increase the chance of such a node being chosen as a relay node, thus increasing SAP. A larger value of h leads to a larger compromised key pool, which again increases the chance of a fabricated node serving as a qualified relay.

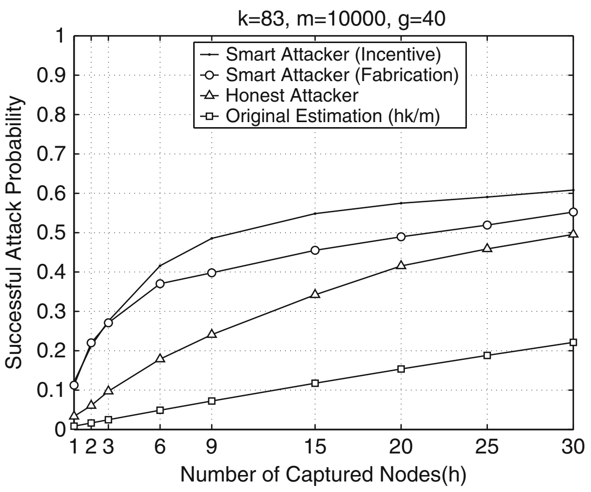


Fig. 23.7 Successful attack probability under different numbers of captured nodes (**h**), for different attack models as well as the estimation in [9].

Figure 23.7 shows the SAP under different numbers of captured nodes, for different kinds of attacks, as well as the estimation based on the result in [9][3]. The number of authorized nodes is fixed at 40. It is clear that the results in [9] significantly underestimate the SAP in mobile networks. With a large enough number of compromised nodes, the SAP can easily reach an unacceptably high value of 50% with all attack models.

Figure 23.9 shows the SAP under different sizes of the preloaded key ring, k, for different attack models as well as the estimation based on the results in [9]. We also plot the link connectivity (i.e., the probability that two neighbor nodes share at least one key, $1 - \delta_m^k$) under different values of k. With the increase in k, the link connectivity increases, as well as the SAP estimation based on the analysis in [9], which is linear in k. On the other hand, the SAPs for all three attack models actually decrease with an increasing k, due to less need of going through a relay to establish a pairwise link. However, they are still much higher than the original estimation of SAP in [9] and the nodes need more memory to store so many keys.

23.5.2 Security Improvement with SP architecture

In this section, we show the security performance of the proposed SP architecture for lightweight ad hoc networks. We focus on the evaluation of the basic probabilistic

[3] When the network is static, an adversary captures h nodes, then its successful attack probability on a link is $1 - \left(1 - \frac{k}{m}\right)^h \approx \frac{hk}{m}$. if $\frac{k}{m}$ is small

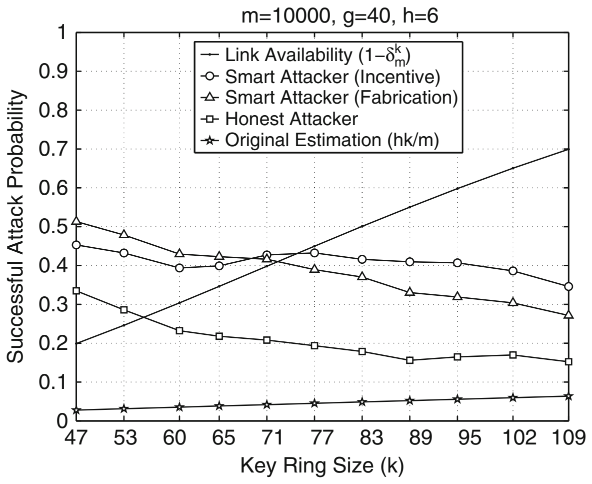


Fig. 23.8 Successful attack probability for different key ring sizes (**k**), for different attack models as well as the estimation in [9].

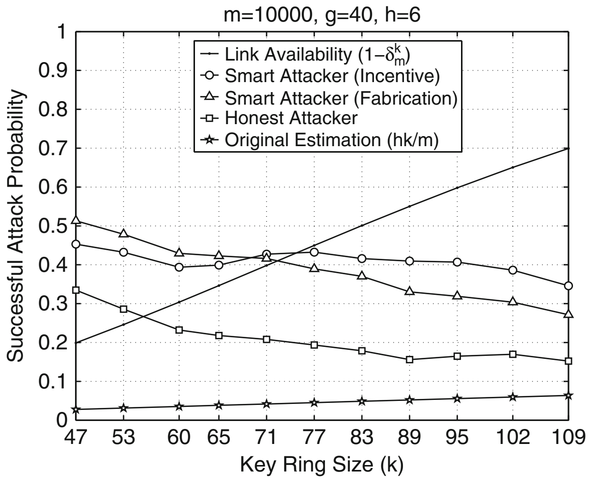


Fig. 23.9 Successful attack probability for different key ring sizes (k), for different attack models, as well as the estimation in [9]

key predistribution approach (the EG scheme) since the deterministic approach (e.g., single common key) is a special case of the probabilistic approach. In addition, many advanced versions of probabilistic key predistribution (e.g., [4, 6]) are also vulnerable to node capture attacks and can benefit from the proposed architecture.

We have run the simulation for a 10×10 grid network, with all nodes assumed to have the same (1 unit) transmission range. A total of 400 nodes are randomly placed in the network. Network-wide SAP is calculated as the fraction of links that can be intercepted by the compromised nodes among all the pairwise links established among the authorized nodes. Again, all simulation results are averaged over 10 sets of random seeds that affect the distributions of the location of each node, the key rings preloaded to nodes, and the relay choices.

We consider several possible attack models depending on whether the SP architecture is used. If every node is equipped with the Sensor-mode SP architecture, the adversary can only launch a *node capture attack*, where the adversary utilizes the captured nodes themselves to intercept pairwise-key establishment. Without the SP architecture, the adversary can further launch *node fabrication attacks* where he can turn the captured nodes into super-nodes by loading each of them with all of the keys from all captured nodes. Each super-node can mimic multiple nodes. A straightforward method to achieve this is to let each super-node stay at its original location but announce the existence of all the captured nodes. The adversary can even make more copies of the super-nodes and deploy them into the network to eavesdrop additional communication. We note that it is difficult to detect the duplication of nodes within the network, since it requires knowledge of the location of each node (possibly using Global Positioning System) and non-trivial communication and memory overhead [26].

Figure 23.10 shows the network-wide SAP under different numbers of captured nodes for different kinds of attacks. "SP" means launching only the node capture attack with the SP architecture. "0 copies" means changing captured nodes into super-nodes (i.e., node fabrication attack) due to the lack of the SP architecture. "x copies" means making x extra copies of these super-nodes elsewhere in the network. We note that the effect of node capture can be serious without SP. When only 3% of the nodes are captured, the SAP for the network will be 9.7% even with "0 copies", and becomes 42.6% if the adversary makes six copies of the captured nodes to cover more area. Whereas the SAP for the nodes with SP is only 2.1%—a reduction by

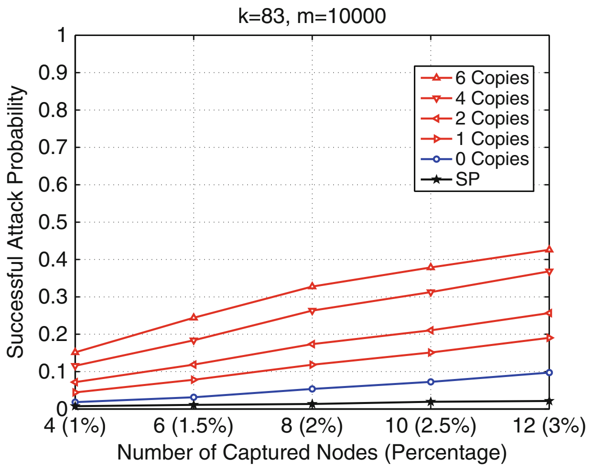


Fig. 23.10 Network-wide successful attack probability under different numbers of captured nodes for different attack models

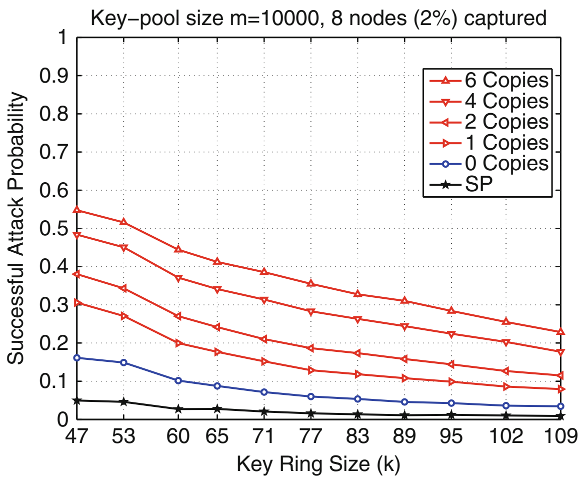


Fig. 23.11 Network-wide successful attack probability for different key ring sizes (k) with different attack models

roughly an order of magnitude. Therefore, SP provides significant benefits in terms of alleviating node fabrication attacks.

Figure 23.11 shows the network-wide SAP under different sizes of the preloaded key ring, k, for different attack models, assuming 2% of nodes have been captured. An increasing value of k has two effects on the network. First, the link connectivity increases; this reduces the probability of two neighboring nodes establishing a pairwise link through a relay node, and thus can improve the network security. Second, each node captured by the adversary contains more keys, which will increase the chance of intercepting the communications on other pairwise links. This is detrimental to the network security. Figure 23.11 shows that the advantage of the first effect dominates and the overall SAP decreases with an increasing value of k. Notice that the SP architecture offers significant advantages over the other schemes for all values of k.

Finally, the single common key scheme also benefits from SP. This is because the adversary, without the ability to learn the common key, can only eavesdrop on the information exchanged within the communication range of the captured nodes.

23.6 Implications to Related Work

23.6.1 Reinforcements on the Basic EG Scheme

Many probabilistic schemes based on the EG scheme have been proposed, e.g., [4, 5, 23, 27, 32]. We show that many of them are also very vulnerable to node capture in the mobile networks, or that the proposed improvements in those schemes can benefit deterministic approaches as well.

In the q-composite key scheme [4], two nodes can only establish a pairwise link between them if they share at least q keys initially (i.e., within the preloaded key rings). The real key for encrypting the communication is a hash result of all q keys. Though this approach can reduce SAP in a static network, it is almost as fragile as the EG scheme in a mobile network, following similar analysis as in Sect. 23.2. In addition, to keep the link connectivity comparable to the EG scheme, the key ring size has to be substantially increased, which means fewer node captures are needed to disclose a sufficiently large key space to the adversary.

The concept of multiple disjoint path key reinforcement was proposed in [4] and [32]. A node will send partial keys to its neighbor through several disjoint paths. The counterpart then regenerates the original key after receiving all these partial keys. A compromised node can only regenerate the key if it intercepts all partial keys. However, this approach requires each node to maintain a global network topology to calculate disjoint paths, and the ability to do source routing. In addition, if virtual node fabrication is possible, there is still a high probability that every path passes through a compromised node. Finally, the approach is also fragile to the attack of deliberate modification by a compromised node along the path, so that the neighbor node cannot successfully regenerate the key.

Some schemes utilize the “partial” deployment information to increase the resilience against node capture (e.g., [5, 23]). In particular, two nodes have a higher probability of sharing keys if they are supposed to be deployed in a group (e.g., in the same geographic area), or have a lower probability if they will be deployed in different groups. Therefore, a node captured in a particular group will have little effect on the security of nodes in other groups. However, the same technique can also be used to enhance the security of a deterministic scheme. For example, a different common key can be assigned to the nodes deployed in the same group, and a node is then equipped with all the keys of the groups it belongs to. Thus capturing one node will not have much adverse effect on the nodes in other groups. Therefore, such improvements do not change the nature of our comparisons of deterministic and probabilistic key management.

23.6.2 Selective Node Capture

So far we have only considered *random node capture*, i.e., the adversary randomly captures nodes to collect sufficient keys to attack the whole network. Another new attack mode, called *selective node capture*, can further weaken the security levels of the probabilistic approaches. As originally proposed in [27], the adversary can listen to the information exchanges when each node tries to identify the keys to be shared with its neighbors. By identifying the key indices in each node and physically locating any node, the adversary can selectively capture nodes with the least overlap in keys. Compared with random node capture, fewer selective node captures are needed to disclose a certain number of unique keys. Although several methods have been proposed to reduce the communication overhead and avoid unnecessary key index disclosure, none of them can completely preclude selective node capture.

One class of key discovery methods is “key indices notification.” A basic approach is that each node announces all of the key indices, with a message size of $O(k)$, as in [1]. Zhu et al. [32] proposed a refined approach that uses a publicly known pseudo-random key index generation function. In the key predistribution phase, the authority’s server chooses a random seed for each node, which calculates a set of k outputs (i.e., key indices) using the key index generation function. The node then uses the random seed as its ID and announces this ID to its neighbors in the key discovery phase. Each of its neighbors can figure out the key indices stored in the node from its ID, since they all have the same key index generation function. As a result, the communication overhead is only of order $O(1)$. However, since the adversary can also figure out all the key index information from the announcements, the scheme is still fragile to selective node capture.

A more complicated challenge-response-like key discovery technique was proposed in [6, 9, 23], where a node can determine whether its neighbor has a particular key if and only if it also has the key. In this scheme, a node announces k challenges separately encrypted by its own k keys. A neighbor node then tries to decrypt the k challenges with its own k keys. Only after the successful decryption of a challenge, can the neighbor node figure out the key from the node who announced the challenge. The process involves k^2 decryption operations and $O(k)$ message exchanges between any two neighbor nodes. To further reduce the computation and communication overhead, Pietro et al. [27] designed a verification function $\Phi(ID\|key)$ using a one-way hash function, like SHA-1, where $\|$ is the concatenation operation. The function $\Phi(ID\|key)$ returns “true” for any argument (i.e., $ID\|key$) with a (pseudo-random) probability of k/m. In the key distribution phase, each node uses $\Phi(ID\|key)$ to test each key k_i from the key pool with its ID and uses that key only if the result is “true”. Therefore, around k keys out of an m sized key pool will be stored on each node. In the key discovery phase, the node just needs to announce its ID, and a neighbor node will execute $\Phi(ID\|k_i)$ k times (i.e., with all its own k keys and the ID it hears) to discover the shared key(s). In short, the procedure involves only k hashing operations and $O(1)$ message exchange between any two neighbor nodes. However, these challenge-response-like methods are still subject to selective node capture. The major difference here is that the selective node capture is only more meaningful than random node capture in sequential node capture mode. That is, in each step, the adversary identifies and captures the node with the fewest keys existing in the compromised key pool. While in “key indices notification” methods, the adversary can identify a set of nodes with the least key overlap and capture them concurrently.

23.7 Key Establishment Approach

As discussed in Sect. 23.1, key pre-distribution schemes have to struggle with the conflicts among node resource limits, desired key-connectivity probability, scalability in network size, and resilience against malicious attacks. Due to the limitation of node memory and computation ability, key predistribution schemes scale poorly in very large networks and the resulting pairwise key-connectivity probability is relatively low. In order to provide an end-to-end key to any communicating node pair, on-demand key establishment becomes a necessary approach. From a security perspective, most key predistribution schemes are designed to protect only the confidentiality of secret keys, while two other security components, integrity and availability, are not accounted for. Key pre-distribution schemes are vulnerable when various attacks occur simultaneously.

To address these issues, a key establishment approach that employs predistributed keys as local link keys has been proposed in [4, 14, 32]. The problem is similar to the verifiable secret sharing [10, 11] in cryptography literature, where most existing algorithms rely on complicated algebraic operations, and thus are unsuitable for ad hoc network applications under computation constraints. In the key establishment approach, to set up an end-to-end secret key between two nodes, the source node generates a set of keying messages, from which a secret key can be derived. Each keying message is sent through a different communication path from the source node to the destination node, which then computes the secret key locally. The transmission is protected by existing link keys at each hop. Since it is difficult to attack a large fraction of keying messages simultaneously in an ad hoc network, the key establishment approach using multi-path is able to guard against various attacks efficiently. In particular, an XOR-based key establishment scheme was proposed in [4, 32], where a secret key is derived by the XOR of all keying messages. This scheme prevents malicious attackers from deriving the secret key if not all keying messages are revealed. In [14], Huang et al. proposed a Reed-Solomon code-based scheme that allows node pairs to derive secret keys when both erasure and modification of keying messages occur. In a closely related problem known as secret sharing [29], it is shown that there exists a scheme to divide a secret into n messages in such a way that the key is easily reconstructable from any $r + 1$ pieces, but even complete knowledge of r pieces reveals no information about the secret. When applied to sensor networks, this technique enables the construction of a key establishment scheme that can guard against both revealing and erasure of keying messages.

However, these key establishment schemes only deal with a subset of the following three attacks, in which malicious nodes (i.e., compromised or fabricated nodes by attackers) can (a) reveal the keying messages passing through them to make secret keys computable to the attackers; (b) erase and not-forward keying messages to prevent other nodes from establishing secret keys; or (c) modify the forwarded keying messages to prevent other nodes from deriving the correct secret keys. These attacks violate the three security properties, confidentiality, availability, and integrity of the keying messages, respectively. To provide a unifying analytical framework for key establishment, in [18] the authors proposed a novel security metric, called the REM resilience vector to quantify the resilience of any key establishment scheme against Revealing, Erasure, and Modification (REM) attacks. Relying on the new security metric, a universal bound on achievable REM resilience vectors was proven for any on-demand key establishment scheme. This bound that characterizes the optimal security performance analytically is shown to be tight, using a REM-resilient key establishment scheme which achieves any vector within this bound. In addition, a low-complexity key establishment scheme which achieves nearly optimal REM attack resilience has been developed in [18].

The remaining parts of this section are organized as follows: In Sect. 23.7.1, we first introduce the analytical framework for key establishment developed in [18], and analyze the security of [4, 14, 29, 32] under the framework. The result in [18] that characterizes the optimal security performance is summarized in Sect. 23.7.2, followed by the low-complexity key establishment scheme in Sect. 23.7.3. Section 23.7.4 contains a simulation, which compares the security performance of all key establishment algorithms.

23.7.1 An Analytical Framework for Key Establishment

Consider a wireless ad hoc network where nodes are not tamper resistant. Compromised or fabricated nodes may reveal all their forwarded keying messages to attackers and also try to disrupt normal key establishment in the network.

In Fig. 23.12, an end-to-end secret key is provided for nodes S and D, who do not share a common key from key predistribution. The procedure is described as follows. After receiving a request message, source node S first employs a network routing protocol and finds m paths (which can be non-disjoint) to the destination node D. Then n keying messages, denoted by $M_1, \dots, M_n$, are generated by the source node and sent to the destination node, each via a different path, i.e., message M_i is send via path $(S, R_{i,1}, R_{i,2}, \dots, D)$. To secure keying messages during transmission, encryptions by existing link keys are performed at each intermediate node before forwarding keying messages, and nodes at the next hop decrypt the messages with the same link keys. More precisely, the following message is sent from node $R_{i,j}$ to node $R_{i,j+1}$:

$$R_{i,j} \rightarrow R_{i,j+1} : E\left[M_i, K_{i,j}^{i,j+1}\right]$$

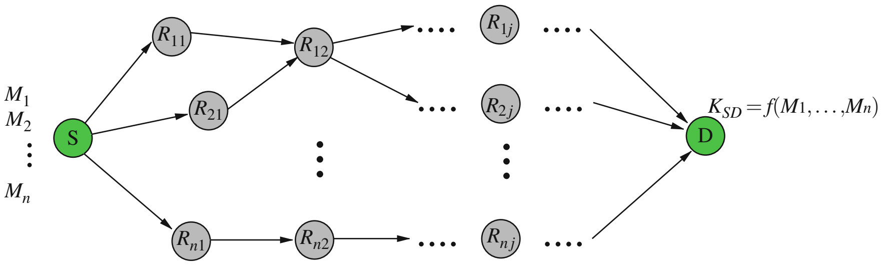


Fig. 23.12 A general key establishment where n messages are sent from the source node S to the destination node D

where $E[\cdot]$ denotes the encryption function and $K_{i,j}^{i,j+1}$ is a link key from key predistribution. Upon receiving the keying messages, node D employs a function $f(\cdot)$ to reconstruct the secret key $K_{SD} = f(M_1, \ldots, M_n)$ for future communication with node S. Since secret keys are set up on demand, the key establishment approach allows rekeying or key refreshing to be easily implemented in wireless ad hoc networks.

In [18], a REM attack is defined as any arbitrary combination of revealing, erasure, and modification attacks. Each type of attack targets at a different security property:

- *Revealing attacks on keying message confidentiality*: Compromised or fabricated nodes reveal to attackers the content of keying messages traveling through them. To quantify the resilience against this attack, we define a threshold value $r \geq 0$, such that if no more than r messages are revealed to attackers, the resulting secret keys remain *completely unknown* even if all attackers collude.

  Definition 1 A secret key generated by a key establishment scheme with function $f(\cdot)$ is *completely unknown* under r revealed messages if

  $$\text{Prob}\left\{f(M_1, \ldots, M_n) = \hat{K} \middle| M_{i_1}, \ldots, M_{i_r}\right\} = \text{Prob}\left\{f(M_1, \ldots, M_n) = \hat{K}\right\}. \tag{23.10}$$

  for any $i_1, \ldots, i_r$ and any choice of key $\hat{K}$.

  Definition 1 implies that revealing any set of no more than r keying messages does not change the original probability distribution of Prob$\{f(M_1, \ldots, M_n)\}$. Thus, attacks obtain 0 information by knowing r out of n keying messages. However, neither S nor D will know if the key is revealed as long as sufficient messages are passed through intact to generate a successful key.
- *Erasure attacks on keying message availability*: In an attempt to prevent the end-to-end secret key from being established, compromised or fabricated nodes make keying messages unavailable to the destination by not forwarding keying messages or jamming the forwarding link. We define $e \geq 0$ to be a threshold such that the secret key can be recovered at the destination node if no more than e messages are erased or dropped.
- *Modification attacks on keying message integrity*: Since complicated authentication methods (e.g., digital signatures using public-key cryptography) are impractical in ad hoc networks, keying messages are subject to modification attacks, in which compromised or fabricated nodes forward modified keying messages to cause confusion. A threshold value $m \geq 0$ is chosen to denote the maximum number of modified messages that can be corrected by a key establishment scheme.

Definition 2 A REM attack in wireless ad hoc networks is defined as any arbitrary combination of the revealing, erasure, and modification attacks, defined above.

Although erasure and modification attacks can also be regarded as transmission erasures and errors from a classical error control coding perspective, this REM attack model is different, because providing confidentiality (which is irrelevant to error control coding applications) jointly with integrity and availability is a must for establishing secret keys. Given that n keying messages are used for establishing a secret key in a key establishment scheme, we quantify its REM attack resilience by introducing a new security metric $(r, e, m)_n$ denoted as a REM resilience vector.

Definition 3 A key establishment scheme using n messages achieves REM resilience $(r, e, m)_n$ if a secret key can be successfully established under no more than e erasure attacks and m modification attacks, and at the same time, the key is completely unknown to attackers for up to r revealed keying messages.

For a key establishment scheme using n keying messages, the set of achievable REM resilience vectors lies in a three-dimensional region, which illustrates security of the particular scheme along three axes: confidentiality, availability, and integrity (see Fig. 23.13).

We can use this unifying framework to analyze the security of any key establishment schemes. In [4, 32], secret keys of length k are derived at destination nodes by the bitwise XOR of all keying messages, each being exactly k bits, i.e.,

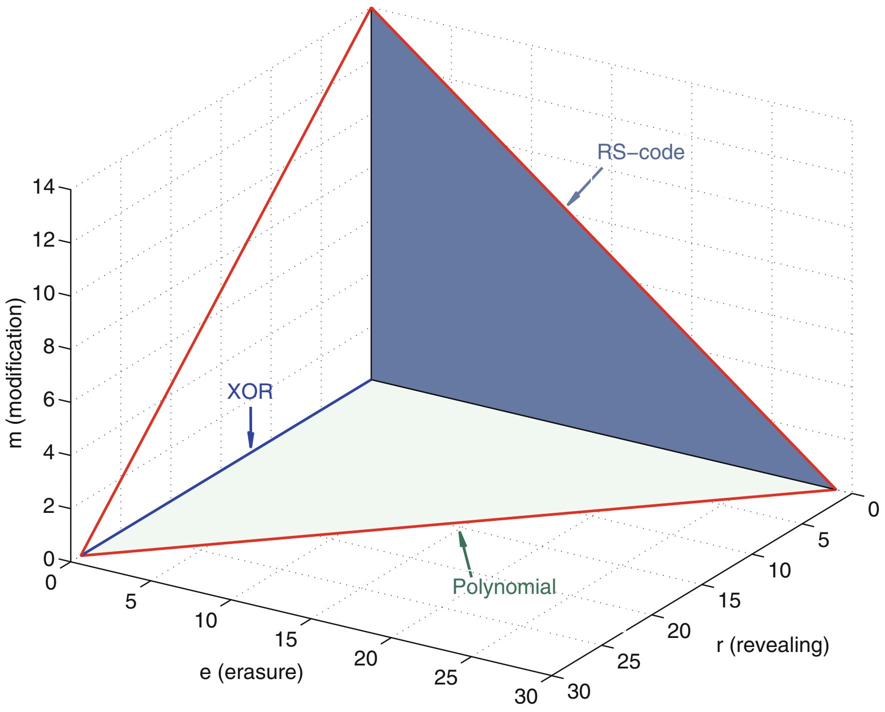


Fig. 23.13 For $n = 30$, this figure plots the 3D optimal REM resilience region (i.e., the tetrahedron defined by $r + e + 2m \leq n - 1$) and 2D sub-planes achieved by previous schemes

$K_{SD} = M_1 \oplus \cdots \oplus M_n$. It is easy to verify that a secret key remains completely unknown if not all keying messages are revealed to attackers. Thus, this scheme achieves REM resilience $(r = n - 1, e = 0, m = 0)_n$. In another scheme based on secret sharing [29], a secret key is regarded as an integer coefficient of a degree t random polynomial in GF_{2^k}, such that it can be recovered from any $t + 1$ evaluations of the polynomial and remains completely unknown if only t evaluations are given. Thus, it achieves $(r = t, e = n - t - 1, m = 0)_n$. By varying the degree t, we denote the set of achievable REM resilience vectors by $(r + e = n - 1, m = 0)_n$.

Another scheme in [14] employs Reed–Solomon (RS) codes to deal with keying message erasures and modifications. The Reed–Solomon codes (RS codes) are non-binary cyclic codes in $GF(2^q)$. RS codes have length $n = 2^q - 1$ with dimension k and minimum Hamming distance $s = n - k + 1$ [21]. Using a secret key of size kq as an input, keying messages are constructed by dividing the output codeword into n pieces, such that the key can be recovered if no more than e and m keying messages are erased and modified respectively, given that $2m + e \leq s - 1$. Since each keying message is a linear combination of the secret key, revealing any keying message makes some choices of keys impossible. Consider a simple scheme with three-bit secret keys $K_{SD} = [b_1 b_2 b_3]$ and a (7,4,3) binary code. If an attack obtains just one bit of the codeword $b_1 \oplus b_2 = 1$, it immediately derives that the secret key cannot be $[00b_3]$ or $[11b_3]$. According to Definition 1, the secret key is not completely unknown to the attacker, and he can remove four possible keys from his entire search space. Thus, we have $r = 0$ for the RS code scheme. Further, by extending this scheme to general non-binary error control codes, a REM resilience of $(r = 0, e + 2m = n - 1)_n$ can be achieved. Table 23.4 summarizes the security analysis of previous key establishment schemes, whose vulnerabilities under REM attacks (i.e., entries with zero resilience) are marked by * in the table.

Table 23.4 Security analysis for key establishment schemes [4, 14, 29, 32]. This shows that these schemes are designed to deal with only a subset of possible attacks

Previous schemes	Resilience vector $(r, e, m)_n$		
	r	e	m
XOR [4, 32]	$r = n - 1$	$e = 0^*$	$m = 0^*$
Polynomial [29]	$r + e = n - 1$		$m = 0^*$
RS code [14]	$r = 0^*$	$2m + e = n - 1$	

23.7.2 *Characterization of Optimal Resilience*

We summarize the results in [18], which analyzes the optimal REM resilience for arbitrary key establishment schemes. For n paths and n keying messages, it is shown that no matter what keying-message construction and function $f(\cdot)$ are used, it is impossible to achieve any REM resilience vector with $r + e + 2m > n - 1$. This result states that $r + e + 2m \leq n - 1$ is a universal upper bound on achievable REM resilience vectors. The upper bound is also tight, as the authors in [18] proposed an optimal key establishment scheme which can achieve any REM resilience vector within this bound.

At first glance, it may appear that both optimality and achievability of bound $r + e + 2m \leq n - 1$ can be readily proved by encoding secret keys using an $(n, r + 1, s)$ linear error control code, since the keys are undecodable from r pieces of output codewords, and a direct application of the Hamming distance gives $2m + e \leq n - r - 1$. However, the result in [18] is much stronger and requires more interesting proofs. First, the definition of security for key establishment requires secret keys to be completely unknown, not even partially decodable. Any piece of output codeword from a simple $(n, r + 1, s)$-encoding reveals certain linear constraints of the secret keys, and thus violates the desired security. Second, our upper bound $r + e + 2m \leq n - 1$ is applicable to any key establishment schemes with an arbitrary keying-message construction and function $f(\cdot)$, while a linear error control code is just one possible approach. The following analysis provides a fundamental limit for the security performance of key establishment, quantified by the proposed REM resilience vector.

Theorem 1 (Optimal Resilience of Key Establishment Approach [18]) *Let each keying message be the same length as the secret key. For n paths and n keying messages, a REM resilience vector $(r, e, m)_n$ can be achieved if and only if $r + e + 2m \leq n - 1$. When the length of keying messages is less than that of the secret key (i.e., length$(M_i) < k$, $\forall i$), it can be proven that a REM resilience $(r, e, m)_n$ can be achieved if and only if* $r + e + 2m \leq n - \left\lceil \frac{k}{\text{length}(M_i)} \right\rceil$.

See Sect. 23.7.5 for proof. Theorem 1 states that for $n > 1$, the set of all achievable REM resilience vectors $(r, e, m)_n$ form a three-dimensional tetrahedron $r + e + 2m \leq n - 1$ as shown in Fig. 23.13, while key establishment schemes in Sect. 23.7.1 only explored certain two-dimensional sub-planes in the tetrahedron: the polynomial-based approach based on [29] achieves $\{r + e \leq n - 1, m = 0\}$, the Reed–Solomon code-based approach in [14] achieves $\{r = 0, e + 2m \leq n - 1\}$, and the XOR-based approach in [4] only achieves a single line $\{r \leq n - 1, e = 0, m = 0\}$. Theorem 1 for key establishment includes all previous results as lower-dimensional special cases.

23.7.3 Low-Complexity Algorithm for Key Establishment

It is shown in [18] that achieving REM resilience vectors on the optimal bound requires multiplications of large integers in GF_p with $p > 2^k$ for constructing keying messages and a complicated sphere decoder. This complexity is prohibitive for wireless ad hoc networks. Therefore, the authors also derived a class of low-complexity key establishment schemes that makes use of binary linear error control codes, and only requires bitwise XOR operations and simple table lookups. The algorithm can achieve a nearly optimal REM resilience. In this section, we first explain the basics of binary linear error control codes, describe the low-complexity algorithm in [18], and then provide a security analysis.

Classical linear coding theory focuses on error correcting. A linear code $\mathcal{C}$ over a finite field with q elements is a linear subspace of the field GF_q^N. If $\mathcal{C}$ is an (n, k, s)-code, then it encodes a vector of length k as a codeword of length n. Let G of size $k \times n$ be the generating matrix for this linear code. Codewords can be obtained by linear combinations of the rows of G, i.e., if $\vec{x}$ is a vector of length k, then $\mathbf{y} = G^T \mathbf{x}$ has length N and is the codeword for $\mathbf{x}$. Parameter s is the distance of the linear code, which is equal to the minimal weight (i.e., number of non-zero components) among all non-zero codewords and measures the error correcting capability of code $\mathcal{C}$. In this chapter, we focus on binary linear codes in GF_2 such that each component is either 0 or 1, although most results can be extended to linear codes in general.

To describe the error correcting procedure, we first introduce the concept of dual code and parity check matrix. The orthogonal complement of $\mathcal{C}$, i.e., the set of all vectors in GF_q^n which are orthogonal to every vector in $\mathcal{C}$, is also a subspace and thus another linear code called the dual code of $\mathcal{C}$, denoted by $\mathcal{C}^{\perp}$. It is easy to see that if $\mathcal{C}$ is an (n, k, s)-code, then $\mathcal{C}^{\perp}$ is an $(n, n-k, s')$-code. A generating matrix, denoted by H, for $\mathcal{C}^{\perp}$ is called a parity check matrix for $\mathcal{C}$ and has size $(n-k) \times n$. A parity check matrix H can be used to recover the codewords of $\mathcal{C}$ because they must be orthogonal to every row of H. Suppose $\hat{\mathbf{y}} = \mathbf{y} + \mathbf{t}$ is a faulty codeword with an error vector $\mathbf{t}$. Then we can compute $r = H\hat{y} = Hy + Ht = Ht$. The vector r is called the syndrome of $\hat{y}$, which voices information about the error vector $\mathbf{t}$, since $H\mathbf{y} = 0$ for all codeword $\mathbf{y} \in \mathcal{C}$. To recover the original codeword $\mathbf{y}$ from the faulty codeword $\hat{\mathbf{y}}$, we only need to store a syndrome table containing all possible syndromes together with corresponding error patterns. In decoding, when $\hat{\mathbf{y}} = \mathbf{y} + \mathbf{t}$ is received, we first calculate the syndrome $\mathbf{r} = H\hat{\mathbf{y}}$, look up the syndrome table with index $\mathbf{r}$ to the error vector $\mathbf{t}$, and then recover codeword $\mathbf{y}$ by $\mathbf{y} = \hat{\mathbf{y}} - \mathbf{t}$.

When both error and erasure occur, the following syndrome decoding procedure for binary linear codes is employed: We first replace the erased coordinates by all zeros and ones and compute two different syndromes (i.e., $\mathbf{r}^0$ and $\mathbf{r}^1$) respectively. After looking up $\mathbf{r}^0$ and $\mathbf{r}^1$ in the syndrome table to obtain two different error vectors $\mathbf{t}^0$ and $\mathbf{t}^1$, the one that contains less number of errors on non-erased coordinates gives us the correct syndrome that should be chosen. More precisely, if $\mathbf{r}^0$ (or $\mathbf{r}^1$ instead) gives less error, then the original codeword can be recovered by inserting 0 (or ones) on the erased coordinates and then abstracting the error vector $\mathbf{t}^0$ (or $\mathbf{t}^1$). In classical coding theory, it has been proven that an (n, k, s)-code is able to correct any e erasures and m errors at the same time, given that $2m + e \leq s - 1$. The following example contains a generating matrix and a parity check matrix for a (8, 2, 5) linear binary code

$$G = \begin{bmatrix} 1\,1\,1\,1\,1\,0\,0\,0 \\ 0\,0\,0\,1\,1\,1\,1\,1 \end{bmatrix}$$

$$H = \begin{bmatrix} 1\,1\,0\,0\,0\,0\,0\,0 \\ 0\,1\,1\,0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,0\,0\,1\,1 \\ 1\,1\,1\,1\,0\,1\,1\,1 \end{bmatrix}$$

For an input vector $\mathbf{x} = [1\ 1]^T$, the corresponding codeword is given by $\mathbf{y} = G^T\mathbf{x} = [1\ 1\ 1\ 0\ 0\ 1\ 1\ 1]^T$. Now, suppose that the fist two bits of $\mathbf{y}$ are erased and the third bit is flipped, i.e., $\hat{\mathbf{y}} = [*\ *\ 0\ 0\ 0\ 1\ 1\ 1]^T$. In order to recover the original codeword from $\hat{\mathbf{y}}$, we compute two syndromes respectively, $\mathbf{r}^0 = [0\,0\,0\,0\,0\,1]^T$ and $\mathbf{r}^1 = [0\ 1\ 0\ 0\ 0\ 1]^T$. By looking up the syndrome table for this (8, 2, 5)-code, we get $\mathbf{t}^0 = [0\ 0\ 0\ 1\ 1\ 0\ 0\ 0]$ and $\mathbf{t}^1 = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0]$. Since $\mathbf{t}^0$ contains two errors on non-erased coordinates, while $\mathbf{t}^1$ contains only one error, we choose all ones on the erased bits in $\hat{\mathbf{y}}$ and subtract $\mathbf{t}^1$ from it. This gives us the correct codeword $\mathbf{y}$. In the next, we generalize this syndrome decoding method and derive an algorithm for secure key establishment. The proposed algorithm not only corrects modifications and erasures, but also makes secret keys completely unknown to attackers.

Now, we summarize the protocol in [18], for key establishment in wireless ad hoc networks—the low-complexity algorithm relying on linear binary codes. The protocol is divided into four phases: *(1) Request and Path-discovery*, *(2) Sending Keying Messages*, *(3) Recovering Key*, and *(4) Verification*. Packets transmitted in the protocol have the structure

ID_1	ID_2	Payload	CmdType

where ID_1 and ID_2 are the IDs of the source node and the destination node, respectively. In phase 1, any standard ad hoc network routing, such as the Zone Routing Protocol [12], is employed to discover n paths, after receiving a request for key establishment. In phase 2, a $(n + 1, t, s)$ error control code is used to generate n keying messages. Let G be a generating matrix for the code

$$G = \begin{bmatrix} g_{01} & g_{02} & \cdots & g_{0t} \\ g_{11} & g_{12} & \cdots & g_{1t} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n1} & g_{n2} & \cdots & g_{nt} \end{bmatrix}_{(n+1)\times t} \tag{23.11}$$

In order to add freshness to the algorithm, the source node constructs t length-k pseudo-random vectors $X_1, \ldots, X_t$ and encodes each column of matrix $[X_1, \ldots, X_t]$ using G:

$$[K_{SD}, M_1, \ldots, M_n]^T = G \cdot [X_1, \ldots, X_t]^T \tag{23.12}$$

where the first row of the output codeword is chosen as a secret key and the $(i+1)$th row as keying message $\mathbb{M}_i$ for $i = 1, \ldots, n$. Since linear binary codes are used, all operations required in this phase are simply binary XORs, denoted by $\oplus$.

Without loss of generality, assume that the last e keying messages are unavailable to the destination node due to erasure attacks and the remaining $n - e$ keying messages contain m faulty ones due to modification attacks. Let H be a parity check matrix of size $(n+1) \times (n+1-t)$ for the generating matrix in (23.11). In phase 3, the destination node implements a key-recovery algorithm based on the syndrome decoding for linear binary codes, as described in Sect. 23.7.3.A. Since the secret key K_{SD} is just the first row of the codeword in (23.12), the algorithm only needs to restore the first row of the codeword, rather than to decode all random vectors $X_1, \ldots, X_t$. In phase 4, the secret key is verified between the source and destination node. Our protocol for establishing a secret key between two nodes S and D is summarized as follows:

Phase 1 *Request and Path-discovery*

1. Node D broadcasts a request for key establishment:

$$D\colon \boxed{D}\boxed{S}\boxed{\text{Void}}\boxed{\text{ReqKey}}$$

2. Node S responses to the request and starts a routing query for node D using the standard Zone Routing Protocol [12].
3. Node S recodes the first n replies to its routing query and prepares n paths to D:

$$(S, R_{i,1}, R_{i,2}, R_{i,3}, \ldots, D), \text{ for } i = 1, \ldots, n$$

Phase 2 *Sending Keying Messages*

1. Node S constructs t length-k pseudo-random vectors $X_1, \ldots, X_t$.
2. The secret key is derived by

$$K_{SD} = (g_{01}X_1) \oplus (g_{02}X_2) \oplus \ldots \oplus (g_{0t}X_t)$$

3. Initialize $i = 1$.
4. Node S generates keying message $\mathbb{M}_i$:

$$\mathbb{M}_i = (g_{i1}X_1) \oplus (g_{i2}X_2) \oplus \ldots \oplus (g_{it}X_t)$$

5. Node S sends $\mathbb{M}_i$ to node $R_{i,1}$ and erases $\mathbb{M}_i$ locally

$$S \rightarrow R_{i,1}\colon \boxed{D}\boxed{R_{i,1}}\boxed{E\left[M_i, K_s^{i,1}\right]}\boxed{\text{EstKey}}$$

6. If $i < n$, let $i = i + 1$ and go to step 4.
7. Node S erases $X_1, \ldots, X_t$ from his memory.
8. Messages are forwarded to node D, for $i = 1, \ldots, n$:

$$R_{i,1} \rightarrow R_{i,2}\text{:} \begin{array}{|c|c|c|c|c|}\hline R_{i,1} & R_{i,2} & E & \left[M_i, K_{i,1}^{i,2}\right] & \text{EstKey} \\ \hline \end{array}$$

$$R_{i,2} \rightarrow R_{i,3}\text{:} \begin{array}{|c|c|c|c|c|}\hline R_{i,2} & R_{i,3} & E & \left[M_i, K_{i,2}^{i,3}\right] & \text{EstKey} \\ \hline \end{array}$$

$$\vdots$$

$$R_{i,j} \rightarrow D\text{:} \begin{array}{|c|c|c|c|c|}\hline R_{i,j} & D & E & \left[M_i, K_{i,j}^{D}\right] & \text{EstKey} \\ \hline \end{array}$$

Phase 3 *Recovering Key*

1. Node D receives at least $n - e$ keying messages $\hat{M}_1, \ldots, \hat{M}_{n-e}$.
2. Define a mask vector A according to the indices of received keying messages: $A_1 = 0$ and

$$A_{i+1} = \begin{cases} 1, \text{ if } \hat{M}_i \text{ is received} \\ 0, \text{ otherwise} \end{cases} \forall i = 1, \ldots, n$$

3. Node D computes a submatrix $\tilde{H}$, consisting of the $n - e$ non-erased rows of H:

$$\tilde{H}_i = H_{i+1}, \text{ for } i = 1, \ldots, n - e$$

4. Node D computes a syndrome perturbation vector $\tilde{r}$ as the XOR of the $e + 1$ erased rows of H:

$$\tilde{r} = H_1 \oplus H_{n-e+2} \ldots \oplus H_{n+1}$$

5. Node D computes $R^0 = \tilde{H}^T \cdot \left[\hat{\mathbb{M}}_1, \ldots, \hat{\mathbb{M}}_{n-e}\right]^T$.
6. Initialize $i = 1$. Let $ADDR$ be the base address of the syndrome table stored at the destination node.
7. Retrieve $\mathbf{t^0}$ from address $ADDR + R_i^0$.
8. Retrieve $\mathbf{t^1}$ from address $ADDR + \left(R_i^0 \oplus \tilde{r}\right)$.
9. The ith bit of K_{SD} is given by

$$K_{SD,i} = \begin{cases} \mathbf{t}_1^{\mathbf{0}}, & \text{if } popcnt(\mathbf{t^0} \bigwedge A) < popcnt(\mathbf{t^1} \bigwedge A) \\ 1 \oplus \mathbf{t}_{\mathbf{1}}^{\mathbf{1}}, & \text{otherwise} \end{cases}$$

10. If $i < k$, let $i = i + 1$ and go to step 5.

Phase 4 *Verifying Key*

1. Node D generates a random message R and computes its hash value $h(R)$.
2. Node D broadcasts a challenge using secret key K_{SD}:

$$D{:}\,\boxed{D}\boxed{S}\boxed{E[(R,h(R)),K_{SD}]}\boxed{\text{GotKey}}$$

3. Node S decrypts $E[(R,h(R)),K_{SD}]]$ using its version of secret key K_{SD} and obtains $\hat{R}$.
4. Node S broadcasts an acknowledgement

$$S{:}\,\boxed{S}\boxed{D}\boxed{\hat{R}}\boxed{\text{ACK}}$$

5. Node D accepts K_{SD} if it receives $\hat{R} = R$.

In Step 5 of phase 3 above, each row of $\left[\hat{M}_1,\ldots,\hat{M}_{n-e}\right]$ is a valid codeword generated by (23.11) with $e+1$ erasures and m modifications. According to the syndrome decoding procedure described in Sect. 23.7.3.A, if we assume that the erased keying messages are all zero vectors, we can compute a syndrome matrix $R^0 = \tilde{H}^T \cdot \left[\hat{M}_1,\ldots,\hat{M}_{n-e}\right]^T$, where each column of R^0 is a syndrome vector. On the other hand, if we assume that the erased keying messages are all one vectors, it is easy to show that the syndrome for the ith row of $\left[\hat{M}_1,\ldots,\hat{M}_{n-e}\right]$ becomes $\tilde{r} \oplus R_i^0$, with $\tilde{r}$ as a perturbation vector defined in Step 4. Therefore, by looking up the syndrome table and comparing resulting error vectors, we can recover the first bit of the secret key, and thereafter bit by bit. In Step 9 of Phase *Recovering Key*, *popcnt* is a population count instruction which counts the number of "1" bits in a word.

The low-complexity key establishment algorithm is able to achieve nearly optimal REM resilience vectors $(r,e,m)_n$ by choosing different linear error control codes. For n paths and n keying messages, the security performance of the algorithm is characterized as follows.

Theorem 2 *(*Resilience of Low-Complexity Key Establishment [18]) *For a linear binary error control code $(n+1,t,s)$ with dual code $(n+1,n+1-t,s')$, the low-complexity key establishment algorithm in [18] achieves a REM resilience vector $(r,e,m)_n$ for $r = s'-2$ and $2m+e = s-2$. In particular, when both codes are maximum distance separable (MDS), the algorithm achieves an optimal REM resilience of $2m+e+r = s+s'-4 = n-3$.*

See Sect. 23.7.6 for proof.

23.7.4 Numerical Simulations

Consider a wireless ad hoc network with $Z = 1000$ nodes, uniformly distributed in a square area of size $L = 100$. We assume that nodes in the neighborhood of communication range $R = 15$ share pre-installed keys with probability p. These pre-installed link keys are used to secure keying messages during transmission. The standard Zone Routing Protocol (ZRP) [12] with a zone radius of $\rho = 2$ hops is employed to discover n paths for each node pair. Due to the page limitation, we focus on security comparisons in this section and do not provide a network-aspect simulation with complexity evaluations. In all numerical examples, compromised nodes are randomly selected from the Z nodes such that the locations of compromised nodes are uniformly distributed in the area. All security performance is evaluated over 40,000 different realizations and node selections.

We define the probability of secure and successful key establishment as the average probability that two nodes can successfully establish a secret key, and at the same time, the secret key remains completely unknown to attackers. For $p = 0.5$ and optimal key establishment, Fig. 23.14 plots the probability of secure and successful key establishment for the use of $n = 1, 5, 10, 20, 30, 40$ keying messages, under REM attacks with equal probability of each type of attack. It can be observed that the optimal key establishment with $n \geq 20$ can safeguard secret keys with a probability of over 80% for as many as 80 (i.e., 8%) malicious nodes, and its security performance benefits from the increase in keying messages as more path diversity is exploited. This figure provides an important benchmark for the design of practical key establishment algorithm for given security requirements and expected fractions of compromised nodes.

For the same network model with $p = 0.5$ and $n = 30$, we compare in Fig. 23.15 the security performance of different schemes: the optimal key establishment algorithm in Sect. 23.7.5, the low-complexity key establishment algorithm in Sect. 23.7.3, key establishment using single path, and the three previous multi-path key establishment schemes. Our low-complexity algorithm proposed in Sect. 23.7.3, which is based on a (31, 11, 11) linear code and its dual (31,20,6) code for achieving resilience $(r = 4, e + 2m = 9)_3 1$, has a performance that is close to the optimal one and is more suitable for practical implementations. This comparison highlights the importance of defending against multiple attacks simultaneously: under REM attacks, the overall security performance of a key establishment algorithm is largely determined by the worst individual-attack resilience (i.e., $\min(r, e, m)$). It also demonstrates the excellent security-complexity properties of our proposed key establishment protocol.

For the same network model with $p = 0.5$ and $n = 30$, we compare the security performance of different schemes: the optimal key establishment algorithm in Theorem 1, the low-complexity key establishment algorithm in [18], key establishment using single path, and the three multi-path key establishment schemes discussed in Sect. 23.7.1. The low-complexity algorithm in [18] has a performance that is close to the optimal one and is more suitable for practical implementations. This comparison highlights the importance of defending against multiple attacks simultaneously: under REM attacks, the overall security performance of a key establishment algorithm is largely determined by the worst individual-attack resilience (i.e., $\min(r, e, m)$).

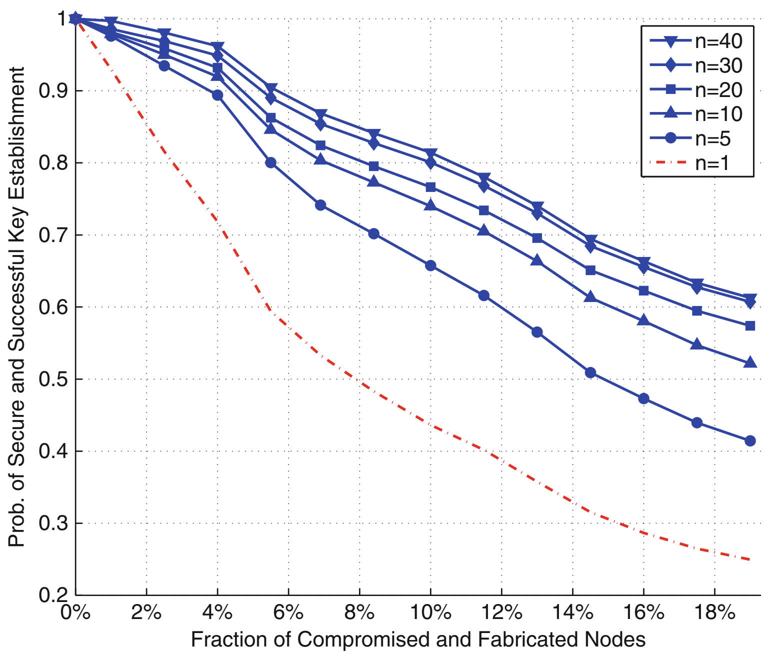


Fig. 23.14 Probability of secure and successful key establishment vs. number of compromised nodes for $n = 1, 5, 10, 20, 30, 40$ keying messages. A diminishing security improvement is observed when more messages are used for key establishment

23.7.5 Proof of Theorem 1

Proof The theorem states that the bound $r + e + 2m = n - 1$ is both optimal and tight. In the following, we start by showing the optimality and then propose a new key establishment scheme to prove the achievability.

To show $r+e+2m = n-1$ is optimal. If $e = m = 0$, then we immediately have $r \leq n - 1$, since the secret key becomes deterministic given all n keying messages. For $e + m > 0$, we denote $[M_1, \ldots, M_n]$ as a *feasible message vector*, in which $M_1, \ldots, M_n$ are a set of allowable keying messages that can be used to establish a secret key $K_{SD} = f(M_1, \ldots, M_n)$. Without loss of generality, we assume that the first r keying messages $M_1, \ldots, M_r$ are revealed to attackers who are able to collude. Then, with this information, the attackers can rule out any feasible message vector whose first r keying messages are not equal to $M_1, \ldots, M_r$. To guarantee that the secret key remains completely unknown, it is necessary that the number of remaining feasible message vectors with the first r messages in common must be no less than 2^k, i.e., the number of all possible secret keys of length k. Formally, if $H(\cdot)$ denotes the entropy function and feasible message vectors are random, we derive

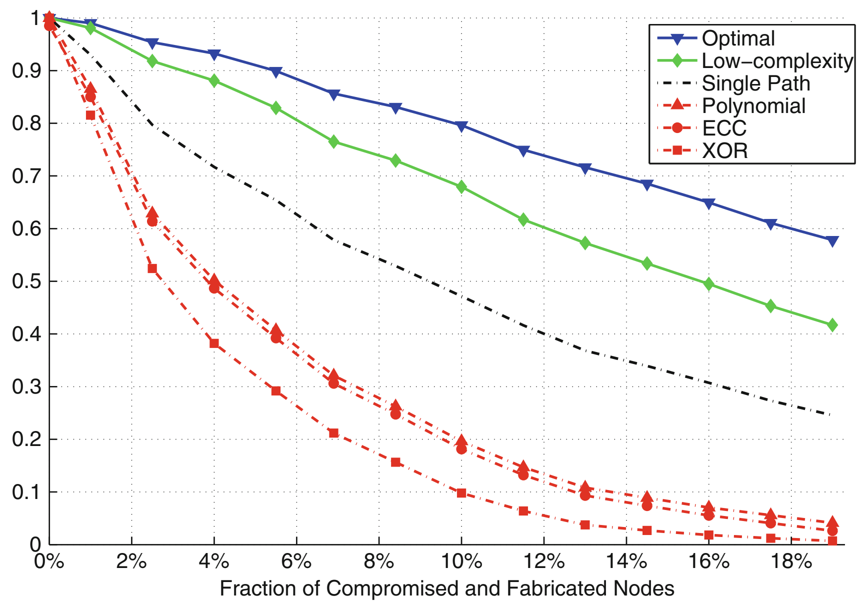


Fig. 23.15 Compare security performance of key establishment schemes under our REM attacks. Our low-complexity algorithm is based on a (1, 31) linear code and its dual (6,20,31) code, and achieves REM resilience $(r = 4, e + 2m = 9)_{31}$

$$\begin{aligned} &H([M_1, \ldots, M_n]|M_1, \ldots, M_r) \\ &\geq H(f(M_1, \ldots, M_m)|M_1, \ldots, M_r) \\ &= H(K_{SD}|M_1, \ldots, M_r) \\ &= H(K_{SD}) = k \end{aligned} \tag{23.13}$$

where K_{SD} is the secret key. The second step is from the information processing inequality and the last step holds because all keys are equally likely due to the definition of completely unknown (23.10). Equation (23.13) implies that with the first r messages fixed, there exists at least 2^k feasible message vectors. These 2^k feasible message vectors are different only in the last $m - r$ messages, each of length k. Thus, the minimum Hamming distance of these feasible message vectors (i.e., the minimum number of different messages in any two feasible message vectors) can be no more than $m - r$. According to error control coding theory, given e erasures and m modifications, two feasible message vectors with a Hamming distance of $m - r$ remain distinct and separable only if

$$2m + e + 1 \leq n - r \quad \Leftrightarrow \quad r + e + 2m \leq n - 1 \tag{23.14}$$

This gives the optimality of bound $r + e + 2m \leq n - 1$.

For achievability of the bound, we propose a new key establishment scheme that achieves any REM resilience vector $(r, e, m)_n$ satisfying the upper bound

$r + e + 2m + 1 = n$. The proposed algorithm for generating n keying messages is similar to the polynomial evaluation used in [29]. However, we employ a different decoding strategy and show that the algorithm can deal with revealing, erasure, and modification attacks at the same time. Let $p > 2^k$ be a prime number. Thus the desired secret key can be regarded as an integer in the field GF_p, i.e., $K_{SD} \in [0, 2^k - 1]$. We generate a random degree r polynomial in GF_p as follows:

$$q(z) = K_{SD} + A_1 z + \cdots + A_r z^{rv} \tag{23.15}$$

where $A_i \in GF_p$ for $i = 1, \ldots, r$ are randomly chosen integers. Then n keying messages are computed by evaluating $q(x)$ at n distinct points for $z = 1, \ldots, n$, i.e.,

$$[M_1, M_2, \ldots, M_n] = [q(1), q(2), \ldots, q(n)] \tag{23.16}$$

Since the polynomial has degree r, it has been shown in [29] that revealing no more than r keying messages would leave the secret key K_{SD} completely unknown. So we only need to show that the destination node can recover key K_{SD} under e erasures and m modifications, given that $2m + e = m - r - 1$. Toward this end, we rewrite (23.16) using a matrix representation:

$$\begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_n \end{bmatrix} = \begin{bmatrix} 1 & 1^1 & 1^2 & \ldots & 1^r \\ 1 & 2^1 & 2^2 & \ldots & 2^r \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & m^1 & m^2 & \ldots & m^r \end{bmatrix} \cdot \begin{bmatrix} K_{SD} \\ A_1 \\ \vdots \\ A_r \end{bmatrix}$$

It is easy to verify that the $n \times (r + 1)$ coefficient matrix (denoted by G) on the right hand side is a Vandermonde matrix, whose any $r + 1$ rows are full rank. Thus, any non-zero vector $\mathbf{x}$ in $GF_p^{(r+1)}$ of size $1 \times (r + 1)$ can be orthogonal to at most r rows of matrix G. We have

$$\forall \mathbf{x} \neq \mathbf{0},\ \text{Hamming}(G\mathbf{x}, \mathbf{0}) \geq n - r \tag{23.17}$$

where $\mathbf{0}$ is a zero vector and Hamming$(\cdot)$ is the Hamming distance function. This implies that matrix G is a generating matrix for a $(n, r + 1, s)$ linear error control code in GF_p with a minimum Hamming distance of at least $n - r$. According to error control coding theory, given that $2m + e + 1 \leq n - r$, any m modifications and e erasures of the keying messages can be corrected at the destination node using a sphere decoding algorithm which finds the closest feasible message vector to the received one [21]. We summarize the optimal key establishment algorithm as follows:

This complete the proof of Theorem 1. ⊠

Optimal Key Establishment Algorithm

1. Source node generates a random key K_{SD} and r random integers $A_1, \ldots, A_r$.
2. Source node generates $\mathbb{M}_i = K_{SD} + A_1 i + \ldots + A_r i^r$ and sends it to destination node, for $i = 1, \ldots, n$.
3. Destination node employs sphere decoding to derive K_{SD} upon receiving the keying messages.

23.7.6 Proof of Theorem 2

Proof We first prove that the proposed algorithm can recover the secret key under e erasure and m modification attacks and then show that attacks have absolutely no information about the secret key with r revealing attacks.

Since each row of the codeword matrix $[K_{SD}, M_1, \ldots, M_n]$ is a valid codeword for the $(n+1, t, s)$ error control code, classical coding theory shows that up to $\left\lfloor \frac{s-1}{2} \right\rfloor$ errors can be corrected by syndrome decoding. In Algorithm 3, we choose the $e+1$ erased keying messages to be all zeros and all ones respectively. Because the error control code is binary, one of the two choices introduces no more than $\left\lfloor \frac{e+1}{2} \right\rfloor$ new errors, and thus leads to no more than $m + \left\lfloor \frac{e+1}{2} \right\rfloor$ errors totally. These errors can be corrected by the syndrome decoding in Algorithm 3 if the following is satisfied:

$$m + \left\lfloor \frac{e+1}{2} \right\rfloor = \left\lfloor \frac{2m+e+1}{2} \right\rfloor \leq \left\lfloor \frac{s-1}{2} \right\rfloor \tag{23.18}$$

This establishes $2m + e \leq s - 2$ as a sufficient condition for recovering secret key K_{SD}.

To show that secret key K_{SD} remains completely unknown to attacks, without loss of generality, we assume that keying messages $M_1, \ldots, M_r$ are revealed to attackers. According to the construction of messages, attackers have $r+1$ equations in the following matrix representation:

$$\begin{bmatrix} K_{SD}^T \\ M_1^T \\ \vdots \\ M_r^T \end{bmatrix} = \begin{bmatrix} g_{01} & g_{02} & \cdots & g_{0t} \\ g_{11} & g_{12} & \cdots & g_{1t} \\ \vdots & \vdots & \ddots & \vdots \\ g_{r1} & g_{r2} & \cdots & g_{rt} \end{bmatrix} \cdot \begin{bmatrix} X_1^T \\ X_2^T \\ \vdots \\ X_t^T \end{bmatrix} \tag{23.19}$$

Because the dual error control code $(n+1, n+1-t, s')$ has distance s', classical coding theory shows that any $s'-1$ rows of the G matrix are linearly independent. Further, s' is upper bounded by $s' \leq t+1$. When $r \leq s'-2$ as claimed in the statement of Theorem 2, we also have $r+1 \leq t$. This implies that the first matrix on the right-hand side of (23.19) is full row-rank.

Thus, when $M_1, \ldots, M_r$ are fixed in (23.19), for each possible choice of secret key K_{SD}, (23.19) defines a system of $r + 1$ linear equations with t unknowns, i.e., $\mathbb{X}_1, \ldots, \mathbb{X}_t$. There exists 2^{t-r-1} possible $\mathbb{X}_1, \ldots, \mathbb{X}_t$ vectors such that (23.19) is satisfied. More precisely, since vectors $\mathbb{X}_1, \ldots, \mathbb{X}_t$ are generated randomly by a uniform distribution, we have

$$\begin{aligned}
&\text{Prob}\left\{K_{SD} = \hat{K} \middle| [M_1, \ldots, M_r] = \hat{M}\right\} \\
&= \frac{\text{Prob}\left\{K_{SD} = \hat{K}, [M_1, \ldots, M_r] = \hat{M}\right\}}{\sum_K \text{Prob}\left\{K_{SD} = K, [M_1, \ldots, M_r] = \hat{M}\right\}} \\
&= \frac{\text{Prob}\left\{[X_1, \ldots, X_t] \in \mathcal{X}_{\hat{K},\hat{M}}\right\}}{\sum_K \text{Prob}\left\{[X_1, \ldots, X_t] \in \mathcal{X}_{K,\hat{M}}\right\}} \\
&= \frac{1}{2^k}
\end{aligned} \tag{23.20}$$

where $\mathcal{X}_{\hat{K},\hat{M}}$ is the set of all $X_1, \ldots, X_t$ satisfying (23.19) for $K_{SD} = \hat{K}$ and $[M_1, \ldots, M_r] = \hat{M}$. Equation (23.20) used the fact that $\left|\mathcal{X}_{\hat{K},\hat{M}}\right| = 2^{t-r}$ for all $\hat{K}$ and $\hat{M}$ and that $\mathbb{X}_1, \ldots, \mathbb{X}_t$ are uniformly distributed. From (23.20), we conclude that given keying messages $\mathbb{M}_1, \ldots, \mathbb{M}_r$, unconditional secrecy as defined in (23.10) is achieved if $v \le s' - 2$.

In addition, according to classical coding theory, for binary error control codes, we have $s + s' = n + 1$ when both the primal and the dual codes are maximum distance separable.Thus, we derive $r + 2m + e = s + s' - 4 = n - 3$, which is the desired result. ⊠

23.8 Concluding Remarks

In this chapter, we discuss key management in lightweight mobile ad hoc networks. Backed up by the large successful attack probabilities computed in this chapter, we show that the probabilistic key predistribution schemes are in fact quite vulnerable to node captures in many practical cases. Considering the large key pool and key ring sizes, complex key predistribution, low network connectivity, and complex pairwise link establishments, the advantage of the probabilistic approach over the deterministic approach is not as much as people have believed. We also generalize the re-examination to other probabilistic key predistribution schemes, including the q-composite key scheme, the multiple disjoint path key reinforcement scheme, and the scheme based on partial deployment information. All of these schemes are vulnerable to node capture in a mobile network. A selective node capture will further weaken the performance of a probabilistic approach.

We then propose two low-cost hardware-based architectures to enhance the security of key management schemes against the attack of sensor node fabrication for a lightweight mobile ad hoc network, which can benefit both probabilistic and deterministic key management.

Finally, we propose a unifying framework for analyzing the security of any key establishment scheme, quantified by a new metric we call a REM resilience vector. A universal bound on achievable REM resilience vectors is derived in closed-form and is shown to be attained by an optimal key establishment algorithm. For practical implementations, we also develop a low-complexity XOR-based key establishment protocol that achieves nearly optimal REM resilience. Our analysis and simulation show that the capability of simultaneously defending against multiple attack classes, critical for the security of wireless ad hoc networks, can indeed be achieved with provable REM resilience and low complexity.

References

1. R. Blom. An optimal class of symmetric key generation system. In *Advanced in Cryptology—Eurocrypt'84*, LNCS, vol. 209, pages, 335–338, 1984.
2. S. Çamtepe and B. Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. In: *European Symposium On Research in Computer Security (ESORICS'04)*, Sophia Antipolis, France, 2004.
3. S. A. Çamtepe and B. Yener. Key distribution mechanisms for wireless sensor networks: a survey. Technical Report TR-05-07, Rensselaer Polytechnic Institute, Computer Science Department, 2005. Available at http://www.cs.rpi.edu/research/pdf/05-07.pdf. Accessed in 2006
4. H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, pages 197–213, Oakland, CA, 2003.
5. W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In: *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, Hong Kong, 2004.
6. W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In: *CCS'03: ACM conference on Computer and communications security*, New York, NY, pages. 42–51, 2003.
7. J. Dwoskin and R. Lee. Hardware-rooted trust for secure key management and transient trust. In: *CCS'07: ACM conference on Computer and communications security*, Alexandria, VA, pages 389–400, 2007.
8. J. Dwoskin, D. Xu, J. Huang, M. Chiang, and R. Lee. Secure key management architecture against sensor-node fabrication attacks. In: *IEEE GlobeCom*, Washington, D.C., pages 166–171, 2007.
9. L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In: *CCS'02: ACM conference on Computer and communications security*, New York, NY, pages, 41–47, 2002.
10. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In: *IEEE Symposium on Foundations of Computer Science*, Los Angeles, CA, Pages 427–438, 1987.
11. M. Fitzi, J. Garay, S. Gollakota, C. Rangan, and K. Srinathan. Round-optimal and efficient verifiable secret sharing. In: *Third Theory of Cryptography Conference*, Volume 3876 of Lecture Notes in Computer Science, pages 329–342, 2006.
12. Z. Haas and M. Pearlman. The performance of query control schemes for the zone routing protocol. In: *IEEE/ACM Transactions on Networking*, 9(4): 427–438, 2001.
13. A. Howard, M. J. Mataric, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In: *Distributed Autonomous Robotic Systems*, Fukuoka, Japan, pages 299–308, 2002.

14. D. Huang and D. Medhi. A byzantine resilient multi-path key establishment scheme and its robustness analysis for sensor networks. In: *19th IEEE International Parallel and Distributed Processing Symposium*, Washington, DC, pages, 4–8, 2005.
15. D. Huang and D. Medhi. Secure pairwise key establishment in large-scale sensor networks: An area partitioning and multigroup key predistribution approach. *ACM Transactions on Sensor Networks* 16:1–34, 2007.
16. D. Huang, M. Mehta, D. Medhi, and L. Harn. Location-aware key management scheme for wireless sensor networks. In: *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, ACM New York, NY, pp. 29–42, 2004.
17. J. Hwang and Y. Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In; *ACM workshop on Security of ad hoc and sensor networks*, Washington, DC, USA, pages, 43–52, 2004.
18. T. Lan, M. Chiang, and R. Lee. Multi-path key establishment against REM attacks in wireless ad hoc networks. In: *IEEE Globecom*, Honolulu, HI, 2009.
19. J. Lee and D. Stinson. Deterministic key predistribution schemes for distributed sensor networks. 11th Annual Workshop on Selected Areas in Cryptography, Waterloo, Ontario, Canada, 2004.
20. R. Lee et al. Architecture for protecting critical secrets in microprocessors. In: *International Symposium on Computer Architecture (ISCA 2005)*, pages, 2–13, 2005.
21. S. Lin and D. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, NJ, USA, 1983.
22. D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In: *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, Washington, DC, pages, 52–61, 2003.
23. D. Liu and P. Ning. Location-based pairwise key establishments for static sensor networks. In: *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, ACM New York, NY, pages. 72–82, 2003.
24. T. Matsumoto and H. Imai. On the key predistribution systems: a practical solution to the key distribution problem. In: *Advances in Sryptology–Crypto'87*, Santa Barbara, CA, 1987.
25. C. S. R. Murthy and B. S. Manoj. *Ad Hoc Wireless Networks: Architectures and Protocols*. Prentice Hall Communications Engineering and Emerging Technologies Series, NJ, USA, 2004.
26. B. Parno, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In: *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 49–63, California, USA, 2005.
27. R. D. Pietro, L. V. Mancini, and A. Mei. Random key-assignment for secure wireless sensor networks. In: *SASN'03: ACM workshop on Security of ad hoc and sensor networks*, New York, NY, pages. 62–71, 2003.
28. S. Seys and B. Preneel. The wandering nodes: Key management for lower-power mobile ad hoc netowrks. In: *IEEE International Conference on Distributed Computing Systems Workshops, ICDCSW'05*, 2005.
29. A. Shamir. How to share a secret. In: *Communications of the ACM*, 1979.
30. S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In: *CCS'03: ACM conference on Computer and communications security*, New York, NY, pages. 62–72,, 2003.
31. S. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. Security and Privacy, 2004. In: *Proceedings*. IEEE Symposium, pages, 259–271, California, USA, 2004.
32. S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach. In: *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03)*, pp. 326–335, 2003.

Chapter 24
Key Predistribution in Wireless Sensor Networks When Sensors Are Within Communication Range

Sushmita Ruj, Amiya Nayak, and Ivan Stojmenovic

Abstract Wireless networks are more vulnerable to security threats than wired networks. Since sensors are resource constrained, the use of traditional cryptographic key management techniques is not practical. Hence keys are distributed in sensor nodes prior to their deployment. This method, called key predistribution, was investigated recently in a number of studies. This chapter restricts the discussion to single-hop networks, where any two sensors are within communication range of each other. The goal is to enable any two sensor nodes to exchange information using their common key, so that other sensors, or an adversary, are unable to decode the message. If two sensor nodes do not share a common key then a path between them, via other sensor nodes, is established, with sensors on the path being able to decode a message and forward it encrypted with a new key. We describe different types of key predistribution schemes for single-hop networks and discuss their merits and demerits in terms of resiliency (impact of node compromises), scalability, connectivity, and memory, computation, and communication resources. Shared-key discovery process should minimize the use of communication bandwidth. We also discuss the identification of compromised nodes and revocation techniques.

24.1 Introduction

Recently, there has been a lot of research in the field of sensor networks. Sensor devices have a wide variety of application both in military and in civilian areas. They are used to collect magnetic, seismic, and acoustic information and sense the temperature and pressure of a given region. Sensors are used in wildlife exploration, ocean water monitoring, plantation monitoring, cold chain management, rescue

S. Ruj (✉)
Department of Electrical and Information Technology, Lund University, P.O. Box 118, S-221 00 Lund, Sweden
e-mail: Sushmita.Ruj@eit.lth.se

Her work is supported by Erasmus Mundus External Cooperation Window Lot 15 India.

operation, vital sign monitoring, and other civilian purposes [76]. They are widely used in military areas for tracking military vehicles, sniper localization, and collecting and transmitting secure information.

Sensor nodes are very small devices with limited memory, battery power, bandwidth, transmission range, and computation power which are scattered in large numbers in the target region and work unattended for large periods of time. Depending upon the battery power, their communication is effective within a region, which is called the radio frequency (RF) region. The RF region is generally a circular region around the sensor node. The radius of this region is called the RF radius. Since they are deployed in large numbers, their cost has to be minimized. An example of a typical sensor is the MICAz mote, which has a low-powered processor with 4 kb of RAM, 512 kb of program memory, an advanced encryption standard (AES) cryptographic hardware and run the TinyOS [1] operating system. The transmitter of the UC Berkley Mica platform has bandwidth of 10 kbps. For more discussion on the resource-constrained nature of sensor networks one may refer to [9, 75].

Data in the sensor nodes deployed in military, health care, or commercial applications need to be securely transmitted. The interception of such data can cause havoc and must thus be prevented. Security in wireless sensor networks (WSN) poses the following challenges [11]: wireless nature of communication, resource limitation on sensor nodes, very large and dense WSN, lack of a fixed infrastructure, unknown network topology prior to deployment, and high risk of physical attacks to unattended sensors.

For security reasons cryptographic keys must be embedded in the sensor nodes which can carry on communication securely. Hence key management becomes of utmost importance in sensor networks. Sensor networks must arrange several types of data packets, including packets of routing protocols and packets of key management protocols. The key establishment techniques must incorporate the following properties [93]:

1. Availability : Ensuring that the service offered by the whole WSN, by any part of it, or by a single sensor node must be available whenever required.
2. Authenticity: Ability for verifying that the message sent by a node is authentic.
3. Confidentiality : The key establishment technique should protect the disclosure of data to unauthorized parties.
4. Integrity : No falsification of data during transmission.
5. Scalability: The key establishment technique must allow for the variation in the size of the network.
6. Flexibility : Key establishment technique should be useful in multiple applications and allow for adding nodes at any time.
7. Non-repudiation: Preventing malicious nodes to hide their activities.
8. Survivability: Ability to provide service in the case of power failure or attacks.
9. Adaptive security service: Ability to change security levels as resource availability changes.

Security protocols in sensor networks have the following constraints and requirements.

1. Memory: The number of keys must be as small as possible.
2. Computation overhead: The amount of calculation for key establishment must be as low as possible.
3. Communication overhead: For key establishment, the nodes must broadcast as little information as possible.
4. Scalability : The system must be able to add more nodes when need arises.
5. Key connectivity: Probability that two sensor nodes share some common key (or share intermediate nodes which share common keys) and thus communicate with each other must be high.
6. Resistance to node fabrication: The schemes must be able to resist node replication to guard against Sybil attacks [25].
7. Revocation: There must be some efficient way to revoke corrupted nodes.
8. Resilience: Once nodes are captured or compromised, the impact of the compromise on the rest of the network must be as low as possible.

One method to establish secret keys is by using public-key protocols. Though there are instances of such schemes [36, 37, 56, 57] using elliptic curves or RSA, such protocols are quite expensive (especially in computation requirements in sensors) and require control and maintenance of keys by base stations and are thus not used much in practice.

Another approach involves the key distribution center (KDC) which is a resource-rich center and acts as a trusted arbiter for key establishment. Examples of such scheme include transport layer security (TLS) [23], security protocols for sensor networks-SPINS [71], and Kerberos [86]. Kerberos is an authentication protocol which is based on Needham and Schroeder's protocol [65]. In Kerberos, the trusted server shares long-lived keys with every node in the network and transmits session keys to sensor nodes on request. This method is extremely expensive for message relay so is not suitable for sensor networks.

The third method is to preload the keys in sensor nodes prior to deployment. This process is called *key predistribution*. There has been an extensive research on key predistribution schemes. Extensive surveys can be found in [2, 11, 77, 93]. In this chapter we describe the problem of key predistribution and discuss different key predistribution schemes, pointing out their merits and demerits.

WSNs use symmetric key mechanism for key establishment, which consists of the following three steps.

1. *Key predistribution*: Preloading keys in sensor nodes prior to deployment. The keys present in a sensor node constitute the key ring (also called key chain) of the sensor. A link exists between two nodes if they share a common key and therefore can communicate directly with each other.
2. *Shared-key discovery*: Communication protocol to find shared key(s) between two communicating nodes.
3. *Path-key establishment*: If a common key does not exists between two communicating nodes, then a path has to be found between the communicating nodes. This path is composed of links among nodes sharing common keys. A path-key

is generated and communicated through the established path. The two communicating nodes communicate using the path-key.

Key predistribution in WSN can be done in any of the following three ways.

1. Probabilistic: Key rings are randomly drawn from a key pool and placed in the sensor nodes. Two nodes communicate with each other with certain probability.
2. Deterministic: Key chains are placed in sensor nodes following some definite pattern.
3. Hybrid: Is a combination of the above two approaches.

In discussing key predistribution it is important to discuss shared-key discovery and path-key establishment, because key predistribution is incomplete without the two. A naive approach to predistribute keys is to use a *single master key* in all the nodes. Thus each node can communicate with every other node in the network using this common key. This scheme is most efficient in terms of storage. However, each node is a single point of failure that brings the whole network down. On the other extreme consider a network of N nodes, each node containing $N - 1$ keys, sharing one key with each of the other $N - 1$ nodes in the network. Such nodes are said to share *pairwise keys*. This guarantees that any node is connected to all other nodes in the network. More importantly, the compromise of one or more nodes does not affect the connection between any other uncompromised nodes. However, since sensors have limited storage, this choice may not be feasible for large sensor networks. Thus there is a trade-off between the storage, key connectivity, and resiliency.

24.1.1 Shared-Key Discovery

When two nodes A and B want to exchange message securely between them, they first use a shared-key discovery algorithm to find some common key between them. Suppose this common key is K. Then node A encrypts a message M and sends $E_K(M)$ to a node B, where $E_K(.)$ is the encryption function using the key K. B on receiving $E_K(M)$, decrypts it using the same common key K such that $D_K(E_K(M)) = M$, $D_K(.)$ is the decryption function using the key K.

There are a few methods of shared-key discovery.

1. *Broadcasting key identifiers*: Two nodes, wishing to communicate, broadcast (without any encryption) the list of key identifiers that each of them possess. Then each node compares the list of identifiers obtained from the other node with its set of identifiers and finds a common key identifier. Communication takes place using the corresponding key. Suppose A has the keys $K_{i_1}, K_{i_2}, \ldots, K_{i_k}$ and B has $K_{j_1}, K_{j_2}, \ldots, K_{j_k}$. So node A creates the list of identifiers $L_A = (i_1, i_2, \ldots, i_k)$ and B has the identifiers $L_B = (j_1, j_2, \ldots, j_k)$, such that $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots < j_k$. Then A broadcasts this list L_A. Upon receiving L_A, B compares L_A with its own list L_B and finds a common key identifier. Suppose $i_a = j_b$. Then communication takes place using

the key $K_{i_a} = K_{j_b}$. The list of identifiers can be sorted in $O(k \log k)$ time. Once the lists are sorted, comparing the lists takes $O(k)$ time. So the computation costs is $O(k \log k)$. Each key identifier can be represented by $O(\log v)$ bits (since there are a total of v keys in the key pool). Hence, to broadcast k identifiers, $O(k \log v)$ bits are required. So the communication overhead is $O(k \log v)$ bits per sensor.
Note: An adversary does not know which key identifier maps to which key (assuming that length of key can be recognized somehow). Unless a node is compromised, the adversary does not know what keys the node contains. However, by knowing the key identifiers in some nodes, an adversary can launch selective node capture attack to speed up full key disclosure (select to capture a node containing the largest number of new keys).

Example 1 Suppose there are 10 keys in the key pool. Let each of the keys be of size 5 bits. These are indexed by $1, 2, 3, \ldots, 10$, such that $K_1 = 10010$, $K_2 = 00011$, $K_3 = 00010$, $K_4 = 10110$, $K_5 = 10111$, $K_6 = 11000$, $K_7 = 11010$, $K_8 = 10100$, $K_9 = 11111$, and $K_{10} = 11011$. Suppose A has the keys $\{K_2, K_3, K_4\}$ and B has the set of keys $\{K_4, K_7, K_{10}\}$. Then A transmits the list of identifiers $L_A = (2, 3, 4)$ and B transmits $L_B = (4, 7, 10)$. So A and B compare the list of keys obtained from each other and finds out that 4 is the common key identifier and encrypts messages using K_4. An adversary eavesdropping on the network knows what keys A and B possess, but not their values. Therefore the adversary also learns that K_4 is the common key between nodes A and B. However, she does not know the value of K_4, because A and B transmit only the key identifiers and not the exact values. If the adversary compromises node A, then she will learn keys in A. She then also learns that the value of the common key between A and B is $K_4 = 10110$. However, she still does not know the values of the other keys that B possesses.

2. *Using challenge–response protocol*: To find one or more shared keys between two nodes, each node has to broadcast a list $\{\alpha, E_{K_i}(\alpha), i = 1, 2, \ldots, k\}$, where α is a challenge. The decryption of $E_{K_i}(\alpha)$ with proper key by the other node would reveal the challenge α and agree on a common key with the broadcasting node. This approach has been adopted in schemes like [19, 33]. Suppose the communicating nodes A and B have the keys $K_{i_1}, K_{i_2}, \ldots, K_{i_k}$ and $K_{j_1}, K_{j_2}, \ldots, K_{j_k}$, respectively. A chooses a challenge α which is a bit string and encodes α with each of the keys that it possesses. A then broadcasts the following information $\{\alpha, E_{K_{i_1}}(\alpha), E_{K_{i_2}}(\alpha), \ldots, E_{K_{i_k}}(\alpha)\}$. Upon receiving this information B decrypts each $E_{K_{i_x}}(\alpha)$ $(x = 1, 2, \ldots, k)$ with its own keys. Suppose $i_A \in \{i_1, i_2, \ldots, i_k\}$ and $j_B \in \{j_1, j_2, \ldots, j_k\}$. Suppose $D_{K_{j_B}}(E_{K_{i_A}}(\alpha)) = \alpha$, then $K_{i_A} = K_{j_B}$ is the common key between A and B. The communication overhead is $O(k \log v)$, where v is the number of keys in the key pool. The calculation of $E_{K_{i_x}}(\alpha), x = 1, 2, \ldots, k$ encryption will require $O(ek)$ time, where e is the encryption/decryption time.
3. *Using pseudo-random sequence generators*: This method was adopted by Park and Blake [68] and Pietro, Macini, and Mei [72]. The key pool (which consists of v keys) is partitioned into k subsets S_i, where $i = 1, 2, \ldots, k$, such that each

set has $t = v/k$ keys. Each key in a subset has an identifier of length $\log t$ bits. The jth key in the key pool S_i is represented by $k_{i,j}$.
A (l, m)-bit generator is a function $f : \mathbb{Z}_2^l \rightarrow \mathbb{Z}_2^m$, that can be computed in polynomial time (as a function of l). The input $s_0 \in \mathbb{Z}_2^l$ is called a *seed* and the output $f(s_0) \in \mathbb{Z}_2^m$ is called the generated bit stream [88]. f is called *pseudo-random bit generator* (PRBG), if no polynomial time adversary can distinguish the output of f from a truly random binary string of length m.
We note that if there are N nodes in the network, then each node identifier can be represented by $l = \lceil \log N \rceil$ bits. Let $m = k \log t$, where $t = v/k$.
The keys are distributed in sensor networks, such that there are k keys in each node, one from each of k key pools $S_1, S_2, \ldots, S_k$. To choose the key chain for a node i, a PRBG function f is applied, which takes as input the $l = \lceil \log N \rceil$ bit identifier of the node i and outputs a m bit string such that, $f(\text{sensor ID}) = j_1||j_2|| \ldots ||j_k$, where j_x is the identifier of the key in the key pool S_x.

Example 2 Suppose there are $N = 60$ nodes in the network and $l = \lceil \log N \rceil = 6$. Let the size of the key chain be $k = 5$. Thus the key pool is partitioned into five subsets S_1, S_2, S_3, S_4, S_5. Let the size of the entire key pool be $v = 20$, thus $t = v/k = 4$. Each key identifier in subset S_i will be represented by $\log t = 2$ bits, therefore possible key identifiers are 00, 01, 10, 11. Consider node $A = 35$. It is represented by the string 100011. A PRBG function f is now applied. Let $f(100011) = 1010110100$. The key identifiers chosen from the subsets S_1, S_2, S_3, S_4, S_5 are represented by 10, 10, 11, 01, and 00, respectively. Thus the keys present in node 35 are $k_{1,2}$, $k_{2,2}$, $k_{3,3}$, $k_{4,1}$, and $k_{5,0}$. Suppose two nodes A and B want to communicate with each other. Given the node id of B, node A can find out the key identifiers that node B has, using the PRBG function f. Then node A compares this list of key identifiers with its own list of identifiers and finds out the common key identifier. The nodes communicate using this common key. Let $B = 33$ be represented by node identifier 100001. Upon receiving key identifier of B, A uses the function f and calculates $f(100001) = 1100100101$. A then derives key identifiers possessed by B: $(1, 3)$, $(2, 0)$, $(3, 2)$, $(4, 1)$, and $(5, 1)$. It compares its own list of identifiers with that of B and finds that $(4, 1)$ is the common key identifier, so communication takes place using the key $k_{4,1}$.

The communication overhead is $O(\log N)$ bits (to broadcast the node identifier). If a node is compromised, all the key identifiers belonging to all the nodes could be extracted by the adversary using the PRBG function that is embedded in the node. An adversary cannot simply learn the key identifiers by eavesdropping on the network. It needs to have access to the PRBG function f to find the identifiers of the keys in each node. While in the first method the entire list of identifiers is broadcasted, in this approach only the node identifier is broadcasted. The communication overhead of this approach is less than in the first one. However, since the list of identifiers are sorted, the time taken for comparing is $O(k \log k)$ in the first approach. In the second one the time taken is in the worst case $O(ke)$.

4. *Using only the identifier of the sensor*: In deterministic methods of key predistribution, nodes broadcast only their node identifiers using which the shared keys

can be calculated. There is no need to exchange key identifiers. The communication overhead is $O(\log N)$ bits per node. This technique is mostly used in deterministic key predistribution schemes like [45, 80]. There is a short algorithm embedded in the sensors which generally runs in time, which is polylogarithmic in N. The algorithm takes the identifier of the broadcasting node B as an input and outputs the common key between itself and B. In this way, no node can know all the key identifiers that any other node possesses. In the case of node compromise, the adversary can only find the common keys between the node it has compromised and any other broadcasting node.

We consider the key predistribution scheme given by Lee and Stinson [45]. This has been described in detail in Sect. 24.6.2. Suppose a network contains $N \leq p^2$ nodes, where p is a prime power. To each sensor node we assign an identifier (i, j) such that $i, j \in \mathbb{Z}_p$. Let the number of keys per node be k. The identifiers of the keys are given by (x, y), where $0 \leq x < k$ and $y \in \mathbb{Z}_p$. A node (i, j) is preloaded with k keys, whose identifiers are $(x, (xi + j) \bmod p)$, where $0 \leq x < k$. Suppose node (i, j) wants to find the key that it shares with a node (i', j'). Shared key identifier is $(x, (xi + j) \bmod p) = (x, (xi' + j') \bmod p)$. Thus $(xi + j) \bmod p = (xi' + j') \bmod p$. Node (i', j') broadcasts its node identifier. Upon receiving it, node (i, j) calculates $x = (i - i')(j' - j)^{-1} \pmod p$, if exists, and then checks if $0 \leq x < k$. If no such x exists, then no shared key exists between nodes (i, j) and (i', j'). Note that the inverse $b = a^{-1}$ satisfies $ba \bmod p = 1$ and can be calculated in $O((\log p)^3)$ time [88].

Example 3 Consider a network containing $N = 25$ nodes. Thus $p = 5$. Let the number of keys in each node $k = 3$. Then the node $(2, 3)$ contains the keys $\{(0, 3), (1, 0), (2, 2)\}$. Suppose node $(i, j) = (2, 3)$ wants to find common key with node $(i', j') = (1, 4)$. Then using the algorithm above, $x = (2-1)(4-3)^{-1} \pmod 5 = 1$ which is less than $k = 3$. Hence the common key identifier between the two nodes is $(1, 0)$. Both nodes have the key determined by $(1, 0)$ and can communicate.

An adversary who compromises node (i, j) learns all its key identifiers and corresponding keys. However, she does not know any of the other identifiers and keys that are present in the other node (i', j'). This approach does not reveal the identifiers in each node and so prevents selective node capture attack (choosing which node to capture in order to learn the most new keys).

In the first approach all the key identifiers are broadcasted, whereas in the third approach only the node identifier is broadcasted and the key identifiers of any node can be calculated from this information. The inherent difficulty with these two approaches is that all the key identifiers are known and this fact can be exploited by an adversary to mount a selective node capture attack. Once an adversary compromises some nodes, it can learn the identifiers of other uncompromised nodes. In the last approach only the common key (if it exists) can be calculated by two nodes, without revealing information about other keys in the nodes.

24.1.2 Network Models

There are several ways of classifying WSN, based upon the type of operation and application [59, 60]. Broadly they can be classified as distributed (homogeneous) or hierarchical (heterogeneous) WSNs. Distributed WSN (DWSN) have no fixed infrastructure, and the network topology is unknown prior to deployment. Sensor nodes are randomly scattered in the target region. Generally all nodes in a DWSN have equal capabilities and are thus homogeneous.

Single-hop networks (called also clique graphs) are the ones in which any two nodes are within communication range of each other. However, two nodes may not share a common key and therefore are forced to communicate through intermediate nodes. A base station or a control node monitors the entire network and receives and processes the information from the sensor nodes. When links joining two nodes without shared key are eliminated, the network becomes multi-hop one. Such network can be modeled as a random graph $G(N, p_c)$ as defined by Erdös and Rényi [32]. A random graph $G(N, p_c)$ is a graph consisting of N nodes and p_c representing the probability of establishing a link between two nodes.

Blackburn and Gerke [4] described random graphs as a uniform random intersection graph $G(N, v, k)$. A uniform random intersection graph $G(N, v, k)$ is a random graph consisting of a set V of N nodes, $|V| = N$. Let M be the set of v colors. We assign a subset $F_a \subset M$ of k distinct colors chosen uniformly and independently at random from M ($|M| = v$) to each node $a \in V$. Two distinct nodes $a, b \in V$ are joined by an edge if and only if $F_a \bigcap F_b \neq \emptyset$. For distinct nodes $a, b \in G(N, v, k)$, the probability that ab is an edge is

$$p_c = 1 - \frac{\binom{v-k}{k}}{\binom{v}{k}} \approx \frac{k^2}{v}$$

This approximation holds because a node a is assigned k colors and the probability that each color is assigned to b is k/v. Therefore in a WSN, a $G(N, v, k)$ is modeled by a $G(N, p_c)$, where $p_c = \frac{k^2}{v}$.

Sensor networks containing pairs of nodes which are outside communication range of each other (although they might share a common key) are multi-hop networks, which can be modeled as *unit disk graphs*. Two nodes are able to send messages to each other (which could be encrypted and therefore received but not understood) if and only if they are within the communication range, which is equal for all sensors. In this chapter we will, however, consider only single-hop networks. We will give a brief overview of the key predistribution schemes for multi-hop networks in Sect. 24.7.

Where a shared key exists between nodes, a secure channel is created and all communications between the nodes are performed using the common key. However, there may exist situations where nodes may not share common keys (as in the scheme of [24] which uses 3- *designs*) or when common shared keys are exposed because of node compromise. In such cases a path needs to be established between nodes. Suppose nodes $A = U_0$ and $B = U_L$ having no common key need to communicate with each other. A establishes communication with some node U_1 through some common key which further establishes communication with U_2 and so on. Let $A, U_1, U_2, \ldots, U_{L-1}, B$ be the path between A and B. Suppose A shares a common key k_1 with U_1, U_1 shares a common key k_2 with U_2, and so on, U_{L-2} shares a common key k_{L-1} with U_{L-1} and U_{L-1} shares a common key k_L with B. To send secure message from A to B, A encrypts it with k_1 and sends it to U_1. U_1 decrypts the message using k_1 and encrypts it using k_2 and sends it to U_2 and the process continues. Ultimately the message reaches B encrypted by key k_L known to B thus B can decrypt it and obtain the original message. The message could be a path-key K between A and B, which can subsequently be used for sending other messages. In this case, the path-key is revealed to the nodes on the path where K was originally communicated. In [29], the path for communicating path-key and messages is found in a breadth first manner.

The problem with this approach is that if any node in the path is compromised, then the path-key is also compromised. To resolve this issue, a *multi-path* key establishment scheme is used. In this technique [19] several paths are considered between the communicating nodes and a hash of all the keys is taken as a common key. For example, let there be n paths $p_1, p_2, \ldots, p_n$ between the nodes A and B. Let the path-keys be $pk_1, pk_2, \ldots, pk_n$. Then the final path-key between the nodes A and B is the hash of all these path-keys and calculated as $hash(pk_1||pk_2||\ldots||pk_n)$. The hash function chosen may be SHA-1. This improves the resiliency of the network.

24.1.3 Performance Measures and Notation

The following performance measures, definitions, and notations are used henceforth in the chapter. Table 24.1 gives a list of notations used throughout this chapter.

1. Storage: The number of keys in the key chain of a sensor node.
2. Connectivity: The probability that two nodes within communication range share a common key or a path-key. It can be further specified as follows (for single-hop networks).

   a. s-connectivity (strict connectivity) which is the probability that two nodes share at least one common key.
   b. p-connectivity (path connectivity) which is the probability that two nodes A and B have a secure path between them (can communicate either directly or via other nodes).

3. Degree of a node: The number of neighbors with which it shares one or more common keys.
4. Average path length: The average number of hops between nodes that do not share a common key with one another.

Table 24.1 Notations

Symbol	Meaning
N	Number of nodes in the network
X	Key pool
$v = \|X\|$	Size of the key pool
U_i	Sensor node i
k	Number of keys possessed by a sensor node
s	Number of compromised sensor nodes
t	Security parameter, such that if t or fewer nodes are compromised, The network remains secure
GF(q)	Finite field containing q elements

5. Scalability: The scope of increasing the size of the network, without redistributing keys in the original network.
6. Resiliency: The measure of the tolerance of the sensor network to node compromise. Different authors have used different measures of resilience. Some of these are

   a. number of keys compromised,
   b. probability that a given link is broken,
   c. probability that a (uncompromised) node is disconnected,
   d. collusion resistance, minimum number of nodes to be compromised to have information about the rest of the network and thus break all links in the network. Network is t-secure, if at least $t + 1$ nodes are to be compromised, to break the security of the network.

   A scheme is called *fully resilient* if the compromise of a few nodes does not affect any link between nodes which are not compromised, for example a pairwise scheme that we discussed in Sect. 26.1.
7. Computation and communication overhead: The computation and communication costs incurred during key establishment.

24.1.4 Identifying Compromised Nodes

There are various ways of dealing with compromised nodes. These fall into the following categories.

1. Detect and tolerate false information introduced by attackers [28, 94]. They, however, are not effective in knowing where false information has been introduced and by whom.
2. The other category is application-driven detection mechanism, which enables sensor nodes to monitor the activities of other nodes nearby. If abnormal activity is observed, then a node raises alert signals to the base station or other nodes, which further detects the compromised nodes. Some examples of these *alert-based* techniques have been presented in [35, 50].

3. Sometimes, however, compromised nodes can raise false alarms to mislead the base station or the other nodes. Compromised nodes may also collude (i.e., they behave consistently and do not raise alerts against one another) and increase their influence on the network. In a recent paper, Zhang et al. [97] present an application-independent framework of accurately identifying compromised sensor nodes. The alert reasoning algorithm also takes care of the fact that compromised nodes may collude sometimes. It is optimal in the sense that it identifies the largest number of compromised nodes without introducing false positives.

In the algorithm proposed by Zhang et al. [97], it is assumed that there exists detection mechanism to enable sensor nodes to observe each other's behavior. Node U_i is called the observer of node U_j, if for each event at U_j, U_i will determine whether U_j has behaved correctly. If U_i believes U_j's behavior is suspicious, then an alert will be raised and sent back to the base station. The base station cannot draw any definite assumption from a single alert, so the alerts are observed over a certain period of time (called *time windows*). Within a given time window, the number of events of U_j is a random variable x with the distribution $f_j(x)$. Given the sensor behavior of U_j, observations of U_i and distribution $f_j(x)$, the expected number of alerts raised by U_i against U_j (when U_i and U_j are both uncompromised), within the time window can be derived. This is compared with the actual alerts raised by U_i against U_j. If this actual number is greater than the expected number, within statistical significance, then it is considered abnormal. A graph called the *observability graph* is constructed, which consists of nodes as vertices. An edge exists between two nodes U_i and U_j, if the two nodes communicate with one another. The set of edges is divided into two classes, normal and abnormal edges depending on the fact that alerts being less than or greater than the estimated values, respectively.

Example 4 We consider the example in [97]. Figure 24.1 shows an observability graph with nine nodes. The normal edges are represented by dotted lines and the abnormal edges by solid lines. Given an observability graph $G(V, E)$, let E_a and E_n be the set of abnormal and normal edges in G, respectively. Then two sensor nodes U_i and U_j are said to be a *suspicious pair* if one of the following is true:

1. $(U_i, U_j) \in E_a$ or $(U_j, U_i) \in E_a$
2. There exists a sensor node U', such that either $(U_i, U') \in E_a$ and $(U_j, U') \in E_n$ or $(U_i, U') \in E_n$ and $(U_j, U') \in E_a$.

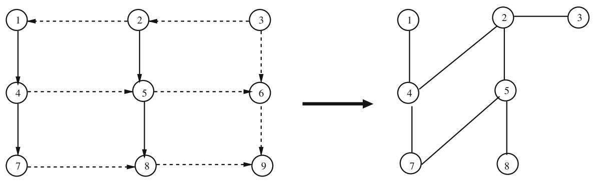


Fig. 24.1 An observability graph (*left*) and the corresponding inferred graph (*right*). The *solid lines* represent abnormal edges and the *dotted lines* represent normal edges in the observability graph

Let $\left(U_1, U_1'\right), \ldots, \left(U_n, U_n'\right)$ be the suspicious pairs derived from an observability graph G. The *inferred graph* of G is an undirected graph $I(V', E')$ such that $V' = \bigcup_{1 \le i \le n} \left\{U_i, U_i'\right\}$ and $E' = \left\{\left(U_i, U_i'\right) | 1 \le i \le n\right\}$. If (U_i, U_j) is a suspicious pair, then at least one of them is compromised. We note that if a pair of nodes is not suspicious, it does not mean that they are both uncompromised. It only means we cannot infer anything about them. Since (2,1) is normal and (2,5) is abnormal, shouldn't (1,5) be also a suspicious pair? It is not always so because it is possible that node 2 is compromised and selectively issues bogus alerts against node 1 but not node 5, even though both nodes 1 and 5 are uncompromised. Similarly, a compromised node may sense data normally but issue bogus alerts or vice versa. Therefore (1, 3) is not a suspicious pair even though (3, 2) is abnormal and (2, 1) is normal. In other words, transitivity does not hold when constructing suspicious pairs.

If all the nodes in the network are compromised, then the base station cannot identify with certainty which nodes are compromised based on alerts. We assume that the total number of compromised nodes is less than C. C is also called the *security estimation* of the network. The security estimation depends on several factors like the attacker's capabilities or the strength of the sensors.

Given an inferred graph $I(V, E)$ and a security estimation C, a *valid assignment* with regard to I and C is a pair (S_g, S_b), where S_g (good set) and S_b (bad set) are two sets of sensor nodes that satisfy all the following conditions:

1. S_g and S_b is a partition of V, i.e., $S_g \bigcup S_b = V$ and $S_g \bigcap S_b = \emptyset$,
2. For any two sensor nodes U_i and U_j, if $U_i \in S_g$ and $U_j \in S_g$, then $\{U_i, U_j\} \notin E$, and
3. $|S_b| \le C$.

For a given graph and a security estimation C, there are several valid assignments. The common nodes S_b in all possible assignments are always compromised and the others may or may not be compromised depending on which assignment is true for the system. Given an inferred graph $I(V, E)$ and a security estimation C, let $\{(S_{g_1}, S_{b_1}), \ldots, (S_{g_n}, S_{b_n})\}$ be the set of all the valid assignments with regard to I and C. $\bigcap_{1 \le i \le n} S_{b_i}$ is called the *compromised core* of the inferred graph I with security estimation C denoted $CompromisedCore(I, C)$. Similarly, $\bigcap_{1 \le i \le n} S_{g_i}$ is called the *uncompromised core* of I with security estimation C, denoted $UncompromisedCore(I, C)$.

The main task is to identify the compromised core. It can be seen that for an inferred graph $I(V, E)$, the size of compromised core is no less than the size of the vertex cover of I. This is because, when we assign an edge (i, j) in I, either $i \in S_b$ or $j \in S_b$ or both $i, j \in S_b$. Thus S_b is essentially the vertex cover of I, denoted by $VC(I)$. This leads to the following theorem.

Theorem 1 [97] *Given an inferred graph $I(V, E)$, let $VC(I)$ be a minimum vertex cover of I. Then the number of compromised nodes is no less than $|VC(I)|$.*

The following theorem will be used in finding the compromised nodes in a network.

Theorem 2 *Given an inferred graph I and a security estimation C, for any node A in I, $A \in CompromisedCore(I, C)$ if and only if $|N_A| + \left|VC\left(I'_A\right)\right| > C$, where N_A is the set of neighboring nodes of A and I'_A is the graph obtained after removing A and its neighbors from I.*

Proof $\Rightarrow$: Suppose there exists some node A satisfying the relation $|N_A| + \left|VC\left(I'_A\right)\right| > C$ and $A \in S_g$ for some assignment (S_g, S_b). Then we will have $N_A \subseteq S_b$. From Theorem 1, we know the minimum number of malicious nodes in I'_A is $\left|VC\left(I'_A\right)\right|$. Thus we have $|S_b| \geq |N_A| + \left|VC\left(I'_A\right)\right| > C$. This is a contradiction since the maximum number of compromised sensor nodes is C. Thus A must be in S_b under any assignment. Therefore $A \in CompromisedCore(I, C)$.

$\Leftarrow$: Now suppose there exists $A \in CompromisedCore(I, C)$ that satisfies $|N_A| + \left|VC\left(I'_A\right)\right| \leq C$. Then we can always construct an assignment of S_b such that $S_b = N_A \bigcup V_{I'_A}$. Then, however, $A \notin S_b$, which is a contradiction, since $A \in CompromisedCore(I, C)$. Therefore if $A \in CompromisedCore(I, C)$ then $|N_A| + \left|VC\left(I'_A\right)\right| > C$. ☒

Now it can be seen clearly that the algorithm to find the $CompromisedCore(I, C)$ reduces to checking for each node A if $|N_A| + \left|VC\left(I'_A\right)\right| > C$. For this we need to calculate the minimum vertex cover, which is an NP complete problem. So we use an approximation algorithm based on maximum matching of G. Finding size of maximum matching and minimum vertex cover is a related problem, and the following result holds: Given an undirected graph G, $M_G \leq VC(G) \leq 2M_G$ where M_G is the size of the maximum matching of G. If $|N_A| + M_{I'_A} > C$, then $|N_A| + \left|VC\left(I'_A\right)\right| > |N_s| + M_{I'_A} > C$.

Using these results the $CompromisedCore(I, C)$ can be efficiently calculated using the Algorithm 7.

Algorithm 7 AppCompromisedCore(I,C)

```
1: $S_b = \emptyset$
2: **for** Each sensor $A$ in $I$ **do**
3:     Let $N_A$ be the neighbors of $A$
4:     Find the size of the maximum matching $M_{I'_A}$ of $I'_A$
5:     **if** $|N_A| + M_{I'_A} > C$ **then**
6:         $S_b = S_b \bigcup \{A\}$
7:     **end if**
8: **end for**
9: **return** $S_b$
```

We consider the graphs in Fig. 24.1. Let the security estimation $C = 3$. Then for node 1, $N_A = 1$ and $M_{I'_A} = 3$. Therefore $N_A + M_{I'_A} \leq 3$, so 1 is not a compromised node. Table 24.2 below shows the values of N_A, $M_{I'_A}$ and states whether or not the node is compromised.

Some finer adjustments are made such that no false positives are included in the set of compromised nodes. This is presented in [97].

Table 24.2 Finding compromised nodes from inferred graph in Fig. 24.1

Node	N_s	$M_{I'_s}$	Compromised or not
1	1	2	Not compromised
2	3	0	Not compromised
3	1	2	Not compromised
4	3	1	Compromised
5	3	1	Compromised
7	2	1	Not compromised
8	1	2	Not compromised

24.1.5 Node and Key Revocation

Compromised nodes can block the normal functioning of the network by injecting false messages, exhausting resources of legitimate nodes, refusing to carry out the protocol steps, and colluding among themselves to compromise more nodes. It is therefore important to design techniques for sensor and key revocation. It is a challenge to design protocols that will revoke nodes compromised by an adversary despite the presence of the adversary in the system. Also the revocation protocols must use only limited computing and communication resources. Revocation protocols are largely dependent on the underlying key predistribution protocols.

There are two types of revocation protocols: centralized and distributed. In the centralized approach, upon detection of a compromised node, base station broadcasts a revocation message to all sensors nodes that the particular node has been compromised and all copies of keys present in the node must be revoked.

In a distributed approach (first introduced by Chan et al. [19]), the revocation decisions are made by the neighbors of the compromised node. The neighboring nodes cast votes against a compromised node. If the number of votes exceeds a certain predefined threshold t, then the node is revoked and this message is broadcasted throughout the network. Distributed approaches require local broadcasts and are not prone to single point of failure as in the centralized approaches. This approach, however, can result in denial-of-service attacks. Revocation also requires each node to maintain a record of which votes have been heard, which demands appropriate storage.

In [19] the neighboring nodes of a compromised node B cast public votes (that is, by flooding). Each node receives hash value of the votes of all the neighbors of B. Once a vote is cast by a neighbor of B, all the other neighbors check the validity of the vote by comparing the hash value of the vote with that already stored in the node. If the number of valid votes exceeds a predefined threshold, then the node is revoked. However, since all the votes that have been cast by other neighbors of B since the beginning of the lifetime of the network must be stored, this method is memory intensive.

In [17], the votes are not flooded as in the previous case. Neighbors of a node perform voting in sessions to agree to revoke the node. The reason for having voting sessions is to assure that revocation is performed within a certain period of time starting from the time of sending the first revocation vote. This prevents a node to store votes for indefinite period of time. In each voting session, a secret sharing scheme is used to tally the revocation vote from neighbors. Each vote is a secret share. When the number of votes are above a threshold, the secret can be recovered from the shares and this information is broadcasted. Such a broadcast means that the node is compromised and should be revoked.

For each voting session during node initialization, a random polynomial of degree t is generated for each node in the network. The coefficients of the polynomial are random uniformly distributed values in a certain range $[0, l-1]$ (l may be 2^{64}). We note that t must be less than the minimum degree of the network graph. We denote the neighbors of node A by N_A. For each session, node A is given a degree t random polynomial $q_A(x)$. For each node $i \in N_A$, a secret share of $q_A(i)$ is generated. This secret share is masked with Mask_A. A delivers this Mask_A to each of its neighbors using the common key that it shares with the respective neighbor. $E_{\text{Mask}_A}(q_A(i))$ is stored in each node i. The mask is used so that only neighbors of A can decrypt the shares of polynomial $q_A(x)$, while a non-neighbor cannot decrypt it. So this scheme works only when nodes share pairwise keys. Each node $i \in N_A$ also stores hash values of secret shares of each of neighbors of A, i.e., $\text{hash}(q_A(j))$, where $j \in N_A$ and $j \neq i$. A hash function $\text{hash}(y) = x$ is a mapping of a large input y which has a long representation to a short representation x, such that the function hash() is one way. This means that it is easy (polynomial time algorithm exists) to calculate $x = \text{hash}(y)$; however, given the value of x, it is difficult to find y, such that $\text{hash}(y) = x$. In other words it is difficult to invert hash(). Each node also stores the hash of the hash value of $q_A(x)$, i.e., $\text{hash}(\text{hash}(q_A(x))) = \text{hash}^2(q_A(x))$.

When a node B wants to revoke a node A, it sends the revocation vote masked with Mask_A, i.e., $E_{\text{Mask}_A}(q_A(B))$. Suppose a node C has collected t revocation votes against A. Using the mask, it decrypts and obtains t values of the revocation shares. For each of the shares, it verifies it with the hash value already in its memory. Once validated, it can calculate the revocation polynomial $q_A(x)$ of node A from the t shares. It then calculates $\text{hash}(q_A(x))$ and broadcasts this value throughout the network. The reason for sending the hash value and not the original polynomial is that it is costly to broadcast a lengthy set of large polynomial coefficients. The other neighbors $i \in N_A$ ($i \neq C$) of A then verify this using the value of $\text{hash}^2(q_A(x))$ stored in their memory and delete all connections with A. Node A is said to be revoked from the network. Instead of storing all the hash values of the neighbors of a node, a Merkle tree [62] can be used. Only a unifying hash value is stored. However, now instead of sending only one voting information, $O(\log k)$ (k is the number of neighbors of a node) information needs to be revealed and also the secret voting value of a node and $\log k$ internal nodes needs to be send. This reduces the storage space.

24.2 Key Predistribution Schemes in WSN

We discuss two approaches which form the basis of several key predistribution schemes. These schemes, though not introduced for WSN, have been modified and suitably applied by several researchers for key predistribution in sensor networks.

24.2.1 Blom's Scheme

Blom [6] proposed a key predistribution scheme that allows any two nodes of a group to find a pairwise key. The security parameter of the scheme is t. As long as no more than t nodes are compromised, the network is secure (the t-secure property). During the pre-deployment phase, the distribution server first constructs a $(t + 1) \times N$ matrix G over a finite field GF(q), q is a prime power, and N is the size of the network. G is public information; any node can know the contents of G, and even adversaries are allowed to know G. Then the base station creates a random $(t + 1) \times (t + 1)$ symmetric matrix D over GF(q) and computes an $N \times (t + 1)$ matrix $A = (D.G)^T$, where V^T is the transpose of V. Matrix D needs to be kept secret and should not be disclosed to adversaries or any node (however, one row of $(D.G)^T$ will be disclosed to each node). Since D is symmetric, $A.G = (D.G)^T.G = G^T.D^T.G = G^T.D.G = (A.G)^T$. Thus $K = A.G$ is a symmetric matrix. K_{ij} (or K_{ji}) is the pairwise key between node U_i and node U_j. To carry out the above computation, nodes U_i and U_j need to compute K_{ij} and K_{ji}, respectively. This can be achieved using the following key predistribution scheme, for $w = 1, \ldots, N$: store the wth row of matrix A and wth column of matrix G at node U_w. Therefore, when nodes U_i and U_j need to find the pairwise key between them, they first exchange their columns of G and then they can compute K_{ij} using their private rows of A. Because G is public information, its columns can be transmitted in plaintext. It has been proven that the above scheme is t-secure if any $t + 1$ columns of G are linearly independent. This property guarantees that no member other than U_i and U_j can compute K_{ij} or K_{ji} if no more than t members are compromised.

Construction for matrix G [27]: We note that if any $t + 1$ columns of G are be linearly independent, then the scheme is t-secure. Since each pairwise key is represented by an element in the finite field GF(q), if the length of pairwise keys is 64 bits, then q can be chosen as the smallest prime number that is larger than 2^{64}. Let a be a primitive element of GF(q) and $N < q$. That is, each nonzero element in GF(q) can be represented by some power of a, namely a^i for some $0 < i \leq q - 1$. The ijth element of G is given by $G_{ij} = a^{i*j}$, where $0 \leq i \leq t$ and $1 \leq j \leq N$ [55]. It is well known that $a^i \neq a^j$ if $i \neq j$ (this is a property of primitive elements). Since G is a Vandermonde matrix, it can be shown that any $t + 1$ columns of G are linearly independent when $a, a^2, a^3, \ldots, a^N$ are all distinct. To store the wth column of G at node U_w, it is only required to store the seed a^w, and any node can regenerate the column given the seed.

24.2.2 Blundo et al.'s Scheme

This scheme was proposed by Blundo et al. [7] and was not originally used for sensor networks. It uses a symmetric bivariate polynomial over some finite field GF(q), i.e., a polynomial $P(x, y) \in \mathrm{GF}(q)[x, y]$ with the property that $P(i, j) =$

$P(j, i)$ for all $i, j \in \mathrm{GF}(q)$. The base station calculates the univariate value of $f_i(y) = P(i, y)$ and gives this polynomial to node U_i. In order to communicate with node U_j, node i computes the common key $K_{ij} = f_i(j) = f_j(i)$; this process enables any two nodes to share a common key. If P has degree t, then each share consists of a degree t univariate polynomial; each node must then store the $t + 1$ coefficients of this polynomial. These are elements of $\mathrm{GF}(q)$, as are the pairwise keys that are established; thus, storing a degree t share requires as much space as storing $t + 1$ keys. We consider the following example.

Example 5 Suppose $q = 7$ and $P(x, y) = x^2 + y^2 + 5xy$. Then $P(x, y) = P(y, x)$ and there are at most $N = 7$ nodes. The polynomials distributed to nodes $U_1, U_2, \dots, U_7$ are

$$f_0(x) = x^2, \quad f_1(x) = x^2 + 5x + 1, \; f_2(x) = x^2 + 3x + 4, \; f_3(x) = x^2 + x + 2,$$
$$f_4(x) = x^2 + 6x + 2, \; f_5(x) = x^2 + 4x + 4, \; f_6(x) = x^2 + 2x + 1$$

For instance, U_2 and U_6 can calculate the common key as $f_2(6) = 6^2+3\times6+4 = 2$ or $f_6(2) = 2^2 + 2 \times 2 + 1 = 2$.

If an adversary captures s nodes, where $s \le t$, then it does not learn any information about keys established between uncompromised nodes; however, if it captures $t + 1$ or more nodes then it can interpolate to compute polynomial P and hence learn all the keys. This follows from the Lagrange's interpolation formula which states that

Theorem 3 *Suppose q is a prime and $y_0, y_1, \dots, y_t$ are distinct elements in $\mathbb{Z}_q$. Suppose $f_0(x)$, $f_1(x)$, …, $f_t(x) \in \mathbb{Z}_q[x]$ are polynomials of degree at most t. Then there exists a unique polynomial $P(x, y) \in \mathbb{Z}_q[x, y]$ of degree at most t in two variables x and y, such that $P(x, y_i) = f_i(x)$, where $0 \le i \le t$. $P(x, y)$ is given by*

$$P(x, y) = \sum_{j=0}^{t} f_j(x) \Pi_{0\le h\le t, j\ne h} \frac{y - y_h}{y_j - y_h}$$

Consider shares $f_1(x) = x^2+5x+1$, $f_2(x) = x^2+3x+4$, and $f_3(x) = x^2+x+2$ in Example 5. Then

$$\frac{(y-2)(y-3)}{(1-2)(1-3)} = 4y^2 + y + 3, \; \frac{(y-1)(y-3)}{(2-1)(2-3)} = 6y^2 + 4y + 4, \; \frac{(y-1)(y-2)}{(3-1)(3-2)}$$
$$= 4y^2 + 2y + 1$$

Hence,

$$\begin{aligned}P(x, y) &= (x^2 + 5x + 1)(4y^2 + y + 3) + (x^2 + 3x + 4)(6y^2 + 4y + 4) \\ &\quad + (x^2 + x + 2)(4y^2 + 2y + 1) \\ &= x^2 + y^2 + 5xy\end{aligned}$$

If an attacker compromises $t + 1$ or more nodes, then she can obtain shares $f_i(x)$ and using Lagrange's interpolation method, calculate the polynomial $P(x, y)$. Then the entire network is compromised.

We will show that if an adversary compromises less than $t + 1$ nodes $c_1, c_2, \ldots, c_t$, then she cannot construct $P(x, y)$ [88]. Let k be the real key, which can be calculated from the compromised nodes and $k^\star$ be an arbitrary key. We show that there is a symmetric polynomial that is consistent with the information that is known from the compromised nodes and such that the secret key associated with the polynomial $f^\star(x, y)$ is $k^\star$. Therefore the adversary cannot rule out any possible values of the key. Let u and v be any two nodes. $f^\star(x, y)$ can be defined as

$$f^\star(x, y) = f(x, y) + (k^\star - k)\Pi_{0 \le i \le t} \frac{(x - c_i)(y - c_i)}{(u - c_i)(v - c_i)}$$

We see that $f^\star(x, y)$ is a symmetric polynomial. We also note that $f^\star(x, c_i) = f(x, c_i) = f_{c_i}(x)$. Finally we see that $f^\star(u, v) = f(u, v) + k^\star - k = k^\star$. These results show that for any possible value of $k^\star$, there is a polynomial $f^\star(u, v)$, such that $f^\star(u, v) = k^\star$ and such that the secret information held by t compromised nodes is unchanged. Thus we conclude that Blundo's key predistribution scheme is unconditionally secure against any adversary who compromises $\le t$ nodes.

24.3 The Basic and *Q*-Composite Schemes

In the next few sections we discuss several key predistribution schemes specifically designed for sensor networks. We assume that the number of nodes in the network is N. Let X be the key pool, such that $|X| = v$, k is the number of keys in each node, and p_c is the probability that two nodes share a common key. Consider the neighbors of a node A which share keys with it. The degree of node A is the number of such neighbors of A. The degree d of a node is the average of the degrees of all the nodes in the network. $d = p_c * (N - 1)$, where N is the number of nodes in the network.

The first random key predistribution scheme for WSN was proposed by Eschenauer and Gligor [33]. This scheme is known as the *basic scheme*. Many predistribution techniques use this as the underlying scheme. The basic scheme consists of three steps: key predistribution, shared-key discovery, and path-key establishment. We have discussed them in Sect. 24.1.

The network can be modeled as a random graph $G(N, p_c)$ as defined by Erdös and Rényi [32], as discussed in Sect. 24.1.2. The event that two nodes do not share any key arises when k keys in the second node are disjoint from the set of k keys in the first node. The probability that this event happens is $\frac{\binom{v-k}{k}}{\binom{v}{k}} = \frac{((v-k)!)^2}{(v-2k)!v!}$. Probability that two nodes share a common key is given by $p_c = 1 - Pr[\text{two nodes do not share any key}] = 1 - \frac{((v-k)!)^2}{(v-2k)!v!}$. For example, if $v = 10,000$ and $k = 75$ then $p_c = 0.5$.

The main advantages of this scheme are that it is flexible, efficient, and easy to implement. However, the main disadvantage is that since the keys are randomly selected from the key pool, all the key identifiers have to be broadcasted as is or by challenge–response protocol. This consumes a lot of bandwidth. Hwang and Kim [41] revisited the random graph theory and used giant component theory by Erdös and Rényi to show that even if node degree is small, most of the nodes in the network can be connected.

A variation of the basic scheme was proposed by Chan et al. [19]. This scheme is called the Q-composite scheme. According to this scheme two nodes can communicate with each other provided they share $q' \geq q$ keys between them. The key predistribution step is the same as in the basic scheme, in which k keys are selected from a key pool X and placed in sensor nodes. Two nodes wishing to communicate either broadcast their key identifiers or use a challenge–response protocol (given in Sect. 24.1.1) to find out the common shared keys. If $q' \geq q$ keys are shared between two nodes, then a link key K is generated as the hash of all shared keys as $K = \text{hash}(k_1||k_2||\cdots||k_{q'})$. The keys are hashed in some canonical order, for example, based on the order in which they occur in the original key pool X.

Let $p(i)$ be the probability that any two nodes have exactly i keys in common. Any node has $\binom{v}{i}$ ways of picking up i keys from the key pool X. After i keys are drawn from the key pool X, there are $2(k-i)$ keys to be drawn from X for the two sensor nodes. This can be done in $\binom{v-i}{2(k-i)}$ ways. These $2(k-i)$ keys must be partitioned into equal parts for each of the nodes. This can be done in $\binom{2(k-i)}{k-i}$ ways. Thus $p(i) = \frac{\binom{v}{i}\binom{v-i}{2(k-i)}\binom{2(k-i)}{k-i}}{\binom{v}{k}^2}$. The probability of establishing a link between two nodes is given by $p_c = 1-(p(0)+p(1)+\cdots+p(q-1))$. For a given key ring of size k, minimum key overlap q and minimum connection probability p, the largest v is chosen, such that $p_c \geq p$.

Suppose the number of compromised nodes is s. Since each key ring is of size k, the probability that a given key is not compromised is $\left(1-\frac{k}{v}\right)^s$. The fraction of keys compromised is $1-\left(1-\frac{k}{v}\right)^s$. Probability that a link which is composed of hash values of i shared keys is compromised is $\left(1-\left(1-\frac{k}{v}\right)^s\right)^i$. The probability of setting up a secure link is $p(q)+p(q+1)+\cdots+p(k)$. So, the probability that an existing link is broken when s nodes are compromised is

$$\sum_{i=q}^{k} \text{Pr(Link is broken} \mid \text{The link has } i \text{ common keys)Pr(Link is has } i \text{ common keys)}$$

$$= \sum_{i=q}^{k} \left(1 - \left(1 - \frac{k}{v}\right)^s\right)^i \frac{p(i)}{p}$$

The Q-composite scheme has very high resiliency when the number of nodes compromised is small. However, as the number of nodes compromised increases, more keys are compromised until nodes share fewer than q keys and hence no link key exists.

The choice of q also depends on the requirements of the network. If q is small, then the initial s-connectivity is higher. However, the network is more vulnerable to node compromise if the compromise takes place during the setup phase. With large q the initial connectivity is low, but the compromise of nodes during the setup phase does not affect the network greatly. However, as the number of compromised nodes increases, the resilience drastically reduces.

24.4 Random Pairwise Schemes

In a pairwise scheme each pair of nodes shares a secret key. For a network consisting of N nodes, each node has $N - 1$ keys, one key shared with each of the remaining $N - 1$ nodes. This becomes bottleneck if sensor memory is limited. We now discuss some pairwise schemes.

24.4.1 Chan–Perrig–Song Scheme

The *random pairwise keys scheme*, proposed by Chan, Perrig, and Song [19] is a modification of the pairwise scheme (discussed in Sect. 24.1), in which not all the $N - 1$ keys are to be stored in the nodes. Let p_c be the probability that two nodes are s-connected. In the setup phase, the base station generates $N = k/p_c$ key identifiers. Each node identifier is matched with k other IDs, and a pairwise key is generated for each of these k nodes. Then this pairwise key is stored in both the node's key rings along with the identifier of the other node. This helps in node-to-node authentication. The node broadcasts its id. By looking at the id, another node can decide if it shares pairwise keys with this node. A cryptographic handshake is then done between the nodes. Revocation of node is done via public voting as discussed in Sect. 24.1.5.

Advantages of the scheme are that it is fully resilient, ensure node-to-node authorization, and are resistant to node replication. Disadvantage of public voting for revocation may lead to DOS attacks. The technique also has reduced scalability. If new nodes are deployed in the network, then keys need to be redistributed in the system.

24.4.2 Liu–Ning–Li Polynomial-Pool-Based Key Predistribution

The pairwise key scheme of Liu and Ning [48] (later extended by Liu et al. [53]) using polynomial pool-based key predistribution is based on the scheme given by Blundo et al. [7]. However, instead of a single polynomial in Blundo's scheme, there exists a set of polynomials. Two sensor nodes share some secret key if the keys are generated from the same polynomial. The polynomial scheme has been used in two different ways: a grid-based predistribution scheme and a hypercube-based key predistribution scheme.

According to the polynomial-based scheme, a key pool is generated which consists of a set of $\mathcal{F}$ bivariate t-degree polynomials $f(x, y) = \sum_{i,j=0}^{t} a_{ij}x^i y^j$ over a finite field GF(q), (where q is a prime number) such that the function has the property that $f(x, y) = f(y, x)$. Generally q has the value $2^8 + 1$ or $2^{16} + 1$. A node U_i is given a set of $\mathcal{F}_i \subseteq \mathcal{F}$ of s' polynomial shares. The polynomial share given to node U_i is $f(i, y)$, where $f \in \mathcal{F}_i$. Two nodes share a common key, if their polynomial shares are drawn from the same polynomial. $f(i, j) = f(j, i)$ is the common key between the nodes U_i and U_j. According to this approach, each key occupies $(t + 1)\log q$ storage space. The s-connectivity is given by

$$p_c = 1 - \frac{(|\mathcal{F}| - s')!^2}{(|\mathcal{F}| - 2s')!(|\mathcal{F}|)!} = 1 - \prod_{i=0}^{s'-1} \frac{|\mathcal{F}| - s' - i}{|\mathcal{F}| - i}$$

The scheme is t collusion resistant meaning that a minimum of $t + 1$ nodes have to be compromised to compromise the entire network. The probability that a key is compromised is given by

$$p_{\text{comp}} = p_c \times P_{cd} + (1 - p_c)\left[1 - \left(1 - p_c'\right)(1 - P_{cd})^2\right] \text{ where}$$
$$P_{cd} = 1 - \sum_{i=0}^{t} \frac{s!}{(s - i)!i!}\left(\frac{s'}{|\mathcal{F}|}\right)^i \left(1 - \frac{s'}{|\mathcal{F}|}\right)^{s-i}$$

where s is the number of compromised nodes, p_c is the fraction of direct links, $1-p_c$ is the fraction of indirect links, and $p_c' = s/N$. They also extended their scheme to n-dimensional hypercube-based scheme. The above is a special case when $n = 2$.

24.4.3 Probabilistic scheme of Zhu et al.

Zhu et al. [100] proposed a scheme to establish pairwise keys using probabilistic key sharing [8] and threshold secret sharing [84]. This scheme enables any two nodes to establish a pairwise key on the fly, without the use of an online key predistribution center. Nodes find their pairwise keys only by knowing the key ids, and no key identifier list is to be broadcasted. Hence the communication overhead is minimized.

Consider two nodes U_i and U_j that want to communicate. The goal of the algorithm is to establish a key, S, that is known exclusively to U_i and U_j. The basic idea underlying the establishment of such a key S is as follows. The sender node splits S into multiple shares using an appropriate secret sharing scheme. The sender then transmits to the recipient node all these shares, using a different *logical path* for each share. A *logical path* is said to exist between nodes if either they already share one or more keys (direct logical path) or they do not share any keys, but can exchange message securely through other intermediate nodes (indirect logical path). The recipient node then reconstructs S after it receives all (or a certain number of) the shares. This is similar to the multipath key reinforcement of Chan et al. [19].

Initially k keys are predistributed in each sensor from a key pool consisting of v keys. During pairwise key establishment, any node U_i randomly generates the secret key S and derives shares $sk_1, sk_2, \ldots, sk_n$ from S. The random strings sk_1, $sk_2, \ldots, sk_{n-1}$ are such that $|sk_1| = |sk_2| = \cdots = |sk_{n-1}| = |S|$ and $sk_n = S \oplus sk_1 \oplus sk_2 \oplus \cdots \oplus sk_{n-1}$, where $\oplus$ is the XOR operation.

Example 6 Consider two nodes A and B. They are connected via nodes C and D such that there are two paths between A and B, i.e., path $p_1 : A - C - B$ and path $p_2 : A - D - B$. Node A generates a random share S and create two secret shares sk_1 and sk_2 of S, such that $|sk_1| = |sk_2| = |S|$ and $S = sk_1 \oplus sk_2$. Then node A sends the share sk_1 encrypted with the common key that it shares with C. C decrypts it and then re-encrypts it using the common key that it shares with B and sends to B. So B decrypts and obtains sk_1. Similarly A sends the secret share sk_2 to B. Now B can obtain S by calculating $S = sk_1 \oplus sk_2$. We note that none of the nodes C or D can calculate S because they do not have both the secret shares sk_1 and sk_2.

The advantages of this approach are that the communication overhead is very low, since only the identifiers of the nodes need to be broadcasted to find the common keys. The disadvantage is that if some node in a path is compromised, creation of shares and pairwise key establishment process has to be started afresh.

24.5 Grid-Based Key Predistribution Schemes

In this section we discuss grid-based key predistribution schemes. In these schemes the nodes/keys are assumed to be arranged logically on a grid. The geometry of the grid is used to preload the keys in the sensors thus giving rise to different deterministic key predistribution schemes.

24.5.1 PIKE Scheme of Chan and Perrig

In many schemes a path between two nodes is established where the trusted intermediary is the base station. This leads to the problem that the nodes close to the base station forward most of the information and continually lose battery power. As a result these nodes die out before the other nodes in the network. To overcome this problem Chan and Perrig proposed the PIKE scheme [18]. PIKE stands for peer intermediaries for key establishment. According to this scheme keys are established between nodes such that other sensor nodes act as trusted intermediaries. PIKE achieves an overhead of $O\left(\sqrt{N}\right)$ and establishes keys between any two nodes regardless of the network topology or node density.

Suppose the maximum number of nodes in the network is N. In PIKE the node identifiers are arranged in a square grid structure having a dimension $\sqrt{N} \times \sqrt{N}$. Each node (x, y) is loaded with secret pairwise keys shared solely with each node in the two sets:
$\{i, y\}$ for all $i \in \left\{0, 1, 2, \ldots, \sqrt{N} - 1\right\}$ and $\{x, j\}$ for all $j \in \{0, 1, 2, \ldots, \sqrt{N} - 1\right\}$.

Each key is unique and shared only between two nodes, hence they are called *pairwise* keys. Each node thus stores $2\left(\sqrt{N} - 1\right)$ keys.

Suppose two nodes U_A and U_B having identifiers (x_A, y_A) and (x_B, y_B), respectively, want to communicate. If U_A and U_B are either in the same row or in the same column, then they share a pairwise key and can thus communicate directly. If they lie in different row and column, then they can establish a communication between each other through two nodes U_1 and U_2, such that U_1 is in the same row as U_A and same column as U_B and U_2 is in the same column as U_A and same row as U_B. Thus for every pair of nodes, there are two intermediaries.

The deterministic nature of key predistribution guarantees that two nodes will be able to establish a common key. The disadvantages of PIKE are that the communication overhead is high. A large fraction of keys do not share common keys and, path must be established through intermediary nodes which may be time consuming.

Sadi et al. [83] proposed another two-dimensional grid-based random scheme based on bivariate polynomials which they called GBR (grid-based random) key predistribution scheme. Mohaisen et al. [64] introduced a three-dimensional grid-based scheme in which each sensor node corresponds to a grid-intersection point. They also use symmetric polynomials for key predistribution.

24.5.2 Liu–Ning–Du Scheme

In many applications, for example, when nodes are scattered from an airship, nodes may be deployed in groups. The sensors belonging to one group are in close proximity to one another. Liu et al. [51, 52] proposed framework of group-based deployment and presented two key predistribution schemes which are very suitable in this framework. The first scheme is a *hash key-based scheme* and the other a *polynomial-based scheme*.

Each group of sensors that are deployed together forms a *deployment group*. Since in this chapter we discuss only single-hop networks, any sensor is in communication proximity of any other, regardless of group belonging. The sensor nodes to be deployed are divided into n groups each consisting of m sensor nodes. A predistribution scheme exists to establish pairwise keys within each group (called *in-group predistribution*), and a key predistribution method is used to establish pairwise keys in different deployment groups (called the *cross-group predistribution*). The in-group instance D_i can be any existing key predistribution technique. The m cross groups $\left\{G'_i\right\}_{i=1,2,\ldots,m}$ are constructed such that

1. each cross-group includes exactly one sensor node from the deployment group, that is, $\left|G'_i \bigcap G_j\right| = 1, i \neq j$
2. there are no common sensor nodes between any two different cross groups, that is, $G'_i \bigcap G'_j = \emptyset$.

In the hash-based scheme before deployment, every sensor node U_i is predistributed with a master key K_i, known only by the base station and the node U_i. Let G be either a deployment group or a cross group. Assume that the node IDs in G have already been sorted in an ascending order. For any sensor node $U_i \in G$, let $\text{Pos}(i)$ be the position of this node in the ordered group G. For any two nodes U_i and U_j ($\text{Pos}(i) < \text{Pos}(j)$) in this group, the value ($\text{Pos}(i)+ \text{Pos}(j)$) is checked. If it is an odd value, node U_j is predistributed $H(K_i||j)$; otherwise, node U_i is predistributed $H(K_j||i)$, where H is a one-way hash function. Due to the group construction method, the positions of a sensor node in its deployment group and cross group, $\text{Pos}(i) = ((i-1) \mod m) + 1$; in the case of the cross group, $\text{Pos}(i) = \lceil i/m \rceil$. Position of any sensor is part of deployment plan and is therefore recorded at sensor. This method is a variant of grid-based scheme, where $k_{i,j}$ is defined as either $H(K_i||j)$ or $H(K_j||i)$.

Example 7 We refer to Fig. 24.2. There are four groups and three cross groups. The groups contain the following node ids: $G_1 = \{1, 2, 3\}$, $G_2 = \{4, 5, 6\}$, $G_3 = \{7, 8, 9\}$, $G_4 = \{10, 11, 12\}$ and the cross groups are $G'_1 = \{1, 4, 7, 10\}$, $G'_2 = \{2, 5, 8, 11\}$, $G'_3 = \{3, 6, 9, 12\}$. For example, node 1 and node 2 are in the same deployment group. In this case, we will simply predistribute the key $H(K_1||2)$ to node 2. Node 1 can calculate the value $H(K_1||2)$ because it has key K_1 and knows the id of node 2. Suppose nodes 3 and 11 want to communicate, then they can do so via nodes 2 or 12, since both 3 and 11 share common keys with both nodes 2 and 12.

The purpose of using a hash function is twofold. It reduces storage space and, more importantly, makes the network more secure. For instance, consider node 2, which receives the value $H(K_1||2)$. Even if an adversary compromises node 2, she will be unable to learn the value of key K_1 that only node 1 possesses.

The polynomial-based technique (also a variant of grid scheme with key defined appropriately) uses Blundo's scheme [7]. The framework is scalable and can be used to improve any existing predistribution schemes. The disadvantages of this scheme are that the probability of secure communication between cross-group neighbors is very low. The scheme is not suitable for networks which have small group size.

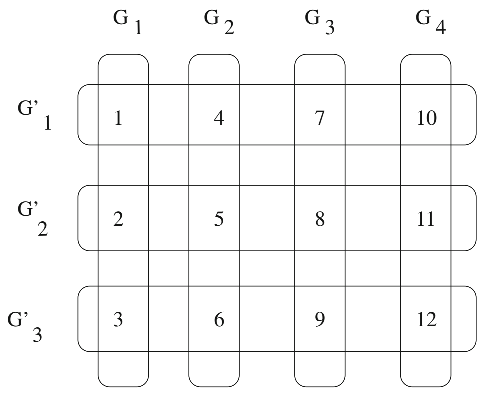


Fig. 24.2 Group-based deployment scheme

24.5.3 Martin–Paterson–Stinson's Improvement of Liu et al.'s Scheme

To alleviate the problems of Liu et al.'s scheme [51, 52] Martin et al. [61] proposed a group-based design using resolvable transversal designs. In their scheme a TD(m, n) is considered.

A *transversal design* [91, Sect. 6.3] TD$(m, \lambda; n)$, with m groups of size n and index λ, is a triple (X, G, A) where

1. X is a set of mn elements,
2. $G = \{G_1, G_2, \ldots, G_m\}$ is a family of m sets (each of size n) which form a partition of X,
3. A is a family of m sets (or blocks) of elements such that each m set in A intersects each group G_i in precisely one element, and any pair of element which belongs to different groups occurs together in precisely λ blocks in A.

When $\lambda = 1$ we can simply denote this by TD(m, n). A TD(m, n) is resolvable if the blocks can be partitioned into parallel classes, such that each point of the design is contained in exactly one block of each class.

The nodes are divided into groups as the previous scheme.

The key predistribution scheme can be described as follows. The key pool consists of $n(m + 1)$ symmetric bivariate polynomials of degree t with coefficients in GF(q), where q may be 2^8 or 2^{16}. There are $N = n^2$ nodes divided into n groups of n nodes each. Each node is given $m + 1$ polynomial shares from the key pool as was done in Blundo's scheme (Sect. 24.2.2), such that the nodes in the same group have shares from only one common polynomial. The nodes in different groups may have shares from at most one common polynomial. We explain the key predistribution scheme using Example 8. The following observations can be made.

1. Each node stores the equivalent of $(m+1)(t+1)$ keys.
2. For each instance of Blundo's scheme (polynomial of Blundo's scheme), there are n nodes possessing shares of that polynomial in that scheme.
3. Probability p_1 that two nodes from different groups share a key is k/n.
4. Probability p_2 that two nodes from the same groups share a key is $\frac{n-1}{n^2/\lambda-1} + \frac{n^2/\lambda-n}{n^2/\lambda-1}mn$.
5. Probability that a uncompromised link is broken when s nodes are compromised is

$$\text{fail}(s) = \begin{cases} 0 & \text{if } s \leq t, \\ 1 - \sum_{i=0}^{t} \frac{\binom{n-2}{i}\binom{n^2-n}{s-i}}{\binom{n^2-2}{s}} & \text{if } s > t \end{cases}$$

Example 8 Let us consider the TD(3,5). Let the set of $mn = 15$ elements be $\{A, B, C, D, E, F, G, H, I, J, K, L, M, N, O\}$. There are $n = 5$ parallel classes. Each element occurs in a parallel class only once. We say group of SN and group of TD to mean the groups of nodes in the sensor network and the groups of the underlying transversal designs, respectively. The groups of the design are

Group 1 of TD : $\{A, B, C, D, E\}$
Group 2 of TD : $\{F, G, H, I, J\}$
Group 3 of TD : $\{K, L, M, N, O\}$

Class 1	Class 2	Class 3	Class 4	Class 5
$\{A, F, K\}$	$\{A, G, M\}$	$\{A, H, O\}$	$\{A, I, L\}$	$\{A, J, N\}$
$\{B, G, L\}$	$\{B, H, N\}$	$\{B, I, K\}$	$\{B, J, M\}$	$\{B, F, O\}$
$\{C, H, M\}$	$\{C, I, O\}$	$\{C, J, L\}$	$\{C, F, N\}$	$\{C, G, K\}$
$\{D, I, N\}$	$\{D, J, K\}$	$\{D, F, M\}$	$\{D, G, O\}$	$\{D, H, L\}$
$\{E, J, O\}$	$\{E, F, L\}$	$\{E, G, N\}$	$\{E, H, K\}$	$\{E, I, M\}$

This gives rise to a network with $n^2 = 25$ nodes deployed in five groups of sensor nodes (SN) of five nodes each. If the nodes have ids $1, 2, \ldots, 25$ then the sensors in each group of sensor nodes will be as follows.

Group 1 of SN	Group 2 of SN	Group 3 of SN	Group 4 of SN
$\{f_1^A, f_1^F, f_1^K, f_1^1\}$	$\{f_6^A, f_6^G, f_6^M, f_6^2\}$	$\{f_{11}^A, f_{11}^H, f_{11}^O, f_{11}^3\}$	$\{f_{16}^A, f_{16}^I, f_{16}^L, f_{16}^4\}$
$\{f_2^B, f_2^G, f_2^L, f_2^1\}$	$\{f_7^B, f_7^H, f_7^N, f_7^2\}$	$\{f_{12}^B, f_{12}^I, f_{12}^K, f_{12}^3\}$	$\{f_{17}^B, f_{17}^J, f_{17}^M, f_{17}^4\}$
$\{f_3^C, f_3^H, f_3^M, f_3^1\}$	$\{f_8^C, f_8^I, f_8^O, f_8^2\}$	$\{f_{13}^C, f_{13}^J, f_{13}^L, f_{13}^3\}$	$\{f_{18}^C, f_{18}^F, f_{18}^N, f_{18}^4\}$
$\{f_4^D, f_4^I, f_4^N, f_4^1\}$	$\{f_9^D, f_9^J, f_9^K, f_9^2\}$	$\{f_{14}^D, f_{14}^F, f_{14}^M, f_{14}^3\}$	$\{f_{19}^D, f_{19}^G, f_{19}^O, f_{19}^4\}$
$\{f_5^E, f_5^J, f_5^O, f_5^1\}$	$\{f_{10}^E, f_{10}^F, f_{10}^L, f_{10}^2\}$	$\{f_{15}^E, f_{15}^G, f_{15}^N, f_{15}^3\}$	$\{f_{20}^E, f_{20}^H, f_{20}^K, f_{20}^4\}$

Group 5 of SN
$\{f_{21}^{A}, f_{21}^{J}, f_{21}^{N}, f_{21}^{5}\}$
$\{f_{22}^{B}, f_{22}^{F}, f_{22}^{O}, f_{22}^{5}\}$
$\{f_{23}^{C}, f_{23}^{G}, f_{23}^{K}, f_{23}^{5}\}$
$\{f_{24}^{D}, f_{24}^{H}, f_{24}^{L}, f_{24}^{5}\}$
$\{f_{25}^{E}, f_{25}^{I}, f_{25}^{M}, f_{25}^{5}\}$

We recall from Sect. 24.2.2 that $f^j(x, y)$ is a symmetric bivariate polynomial of degree t with co-efficients in GF(q), where q is a prime power. The polynomial is evaluated at $x = i$ and the share $f^j(i, y)$ is given to node i. For ease of representation we refer to $f^j(i, y)$ by f_i^j. So in this example, there are $n(m+1) = 20$ symmetric bivariate polynomials $f^A(x, y), f^B(x, y), \ldots, f^O(x, y)$ and $f^1(x, y), f^2(x, y), \ldots, f^5(x, y)$. Each node receives shares of four such polynomials. The shares are univariate polynomials of degree t. For example, node 11 receives and stores the four univariate polynomials $f^A(11, y)$, $f^H(11, y)$, $f^O(11, y)$, and $f^3(11, y)$, which have been represented as f_{11}^A, f_{11}^H, f_{11}^0, and f_{11}^3, respectively. If $f^j(x, y)$ has degree $t = 2$ (say), then each node stores eight values in GF(q) or an equivalent of eight keys.

Suppose nodes 3 and 17 want to communicate. They find out which polynomials they have in common and find that they have shares of the same polynomial $f^M(x, y)$. Node 3 has the univariate polynomial $f_3^M = f^M(3, y)$ and node 17 has the univariate polynomial $f_7^M = f^M(17, y) = f^M(y, 17)$. So the common key between nodes 3 and 17 is $f^M(3, 17)$, which can be calculated by both 3 and 17. If two nodes A and B do not share a common polynomial they do so via other nodes which have polynomials common to A and B. If nodes 3 and 9 want to communicate, they can do so via node 8. In this case $f^C(3, 8)$ is the common key between the nodes 3 and 8 and $f^2(8, 9)$ is the common key between nodes 8 and 9.

The disadvantage with this scheme is that the resolvable designs can be applied where such designs exists. No particular algorithm is given for the construction of such designs for a given set of parameters.

24.6 Key Predistribution Using Combinatorial Structures

Mitchell and Piper [63] were the first to apply combinatorial designs in key distribution in networks. Combinatorial designs were used for the first time in key predistribution by Çamtepe and Yener [10, 12]. Before discussing the key predistribution schemes in WSN, define set systems or designs. More on designs can be found in [87, 91]. A *set system* or *design* [45] is a pair $(X, \mathcal{A})$, where $\mathcal{A}$ is a set of subsets of X, called *blocks*. The elements of X are called elements. The *degree* of a point $x \in X$ is the number of blocks that contain x. $(X, \mathcal{A})$ is regular of degree r if all points have the same degree r. The size of the largest block is called *rank*. Rank is denoted by k. A BIBD($v, b, r, k; \lambda$) is a *design* which satisfies the following conditions:

1. $|X| = v$, $|\mathcal{A}| = b$,
2. The set system is regular of degree r,
3. All sets have the same size equal to k,
4. Each pair of elements in X is contained in exactly λ blocks in A.

Example 9 Consider the following set system $(X, \mathcal{A})$, such that $X = \{0, 1, 2, 3, 4, 5\}$, $\mathcal{A} = \{\{0, 1, 2\}, \{0, 1, 3\}, \{0, 2, 4\}, \{0, 3, 5\}, \{0, 4, 5\}, \{1, 2, 5\}, \{1, 3, 4\}, \{1, 4, 5\}, \{2, 3, 4\}, \{2, 3, 5\}\}$. Here $v = 6$, $b = 10$, $r = 5$, $k = 3$, and $\lambda = 2$. The above example is a BIBD(6,10,5,3;2).

The mapping from set system to key predistribution is done in the following way.

Design parameters	Network parameters
The set of elements X	The set of key identifiers in the key pool
Block	Sensor node
Rank k	Number of keys per node
Degree r	Number of blocks in which a given key is present

The set of elements X represents the key identifiers (ids). Each block represents one sensor node. The subsets of keys denote the key chains. If an element x belongs to block $A \in \mathcal{A}$, then the key having identifier x is placed in the block A. Thus our sensor network consists of a key pool of size $|X|$ and a maximum of $|\mathcal{A}|$ nodes.

Different authors have used different combinatorial structures in devising key predistribution schemes. Key predistribution using combinatorial designs has several advantages. Resiliency can be improved by properly choosing the design. Some designs result in sensor networks where every pairs of nodes share key. The most important point in favor of using combinatorial designs is that because of the pattern inherent in the designs efficient shared-key discovery algorithms can be devised.

The significance of the parameter λ in a BIBD design had not been identified earlier. It is being investigated in [79]. Let us consider a $(v, b, r, k; 2)$ design. Thus $\lambda = 2$. The network arising from this design consists of nodes such that any pair of keys occur in exactly two nodes. So a pairwise key can be constructed between two nodes in the following way. Suppose K_1 and K_2 are two keys which occur only in the nodes A and B. We create a pairwise key $K_{AB} = \text{hash}(K_1||K_2)$. The key predistribution scheme combines the merits of a deterministic scheme as well provides node-to-node authentication (as in pairwise schemes). A closer look reveals that this is essentially a two-composite scheme. It is yet to be verified how well this scheme performs compared to other existing schemes.

24.6.1 Çamtepe and Yener's Scheme

Çamtepe and Yener [10, 12] were the first to use combinatorial designs for key predistribution in WSN. They used projective planes and generalized quadrangles for predistribution. We note that a finite projective plane $PG(2, q)$ (q is a prime power) is the symmetric BIBD, BIBD($q^2 + q + 1, q^2 + q + 1, q + 1, q + 1, 1$). This results in a network with $q^2 + q + 1$ nodes, each having $q + 1$ keys. The size of the key pool is $q^2 + q + 1$. Let the nodes (or blocks) be indexed by (a, b, c) where $a, b, c \in$ GF(q). Nodes are given by the identifiers $(1, b, c)$, $(0, 1, c)$, and $(0, 0, 1)$, where $b, c \in$ GF(q). Thus there are a total of $q^2 + q + 1$ nodes. Similarly the keys are indexed by (x, y, z) where $x, y, z, \in$ GF(q). The identifiers of the keys are given by $(x, y, 1)$, $(x, 1, 0)$, and $(1, 0, 0)$, where $x, y \in$ GF(q). So there are a total of $q^2 + q + 1$ keys (or elements). A key (x, y, z) is assigned to node (a, b, c) if $ax + by + cz = 0$.

Example 10 Let $q = 5$. There are 31 nodes, each node containing six keys. The total number of keys is 31. The nodes are represented by $(1, b, c)$ for $b, c \in \{0, 1, 2, 3, 4\}$, $(0, 1, c)$ for $c \in \{0, 1, 2, 3, 4\}$, and $(0, 0, 1)$. The keys have identifiers $(x, y, 1)$ for $x, y \in \{0, 1, 2, 3, 4\}$, $(x, 1, 0)$ for $x \in \{0, 1, 2, 3, 4\}$, and $(1, 0, 0)$. The distribution of keys for each node is shown in Table 24.3.

Table 24.3 Distribution of keys in nodes

Node	Keys
(1, 0, 0)	(0, 0, 1), (0, 1, 1), (0, 2, 1), (0, 3, 1), (0, 4, 1), (0, 1, 0)
(1, 0, 1)	(4, 0, 1), (4, 1, 1), (4, 2, 1), (4, 3, 1), (4, 4, 1), (0, 1, 0)
(1, 0, 2)	(3, 0, 1), (3, 1, 1), (3, 2, 1), (3, 3, 1), (3, 4, 1), (0, 1, 0)
(1, 0, 3)	(2, 0, 1), (2, 1, 1), (2, 2, 1), (2, 3, 1), (2, 4, 1), (0, 1, 0)
(1, 0, 4)	(1, 0, 1), (1, 1, 1), (1, 2, 1), (1, 3, 1), (1, 4, 1), (0, 1, 0)
(1, 1, 0)	(0, 0, 1), (1, 4, 1), (2, 3, 1), (3, 2, 1), (4, 1, 1), (4, 1, 0)
(1, 1, 1)	(0, 4, 1), (1, 3, 1), (2, 2, 1), (3, 1, 1), (4, 0, 1), (4, 1, 0)
(1, 1, 2)	(0, 3, 1), (1, 2, 1), (2, 1, 1), (3, 0, 1), (4, 4, 1), (4, 1, 0)
(1, 1, 3)	(0, 2, 1), (1, 1, 1), (2, 0, 1), (3, 4, 1), (4, 3, 1), (4, 1, 0)
(1, 1, 4)	(0, 1, 1), (1, 0, 1), (2, 4, 1), (3, 3, 1), (4, 2, 1), (4, 1, 0)
(1, 2, 0)	(0, 0, 1), (1, 2, 1), (2, 4, 1), (3, 1, 1), (4, 3, 1), (3, 1, 0)
(1, 2, 1)	(0, 2, 1), (1, 4, 1), (2, 1, 1), (3, 3, 1), (4, 0, 1), (3, 1, 0)
(1, 2, 2)	(0, 4, 1), (1, 1, 1), (2, 3, 1), (3, 0, 1), (4, 2, 1), (3, 1, 0)
(1, 2, 3)	(0, 1, 1), (1, 3, 1), (2, 0, 1), (3, 2, 1), (4, 4, 1), (3, 1, 0)
(1, 2, 4)	(0, 3, 1), (1, 0, 1), (2, 2, 1), (3, 4, 1), (4, 1, 1), (3, 1, 0)
(1, 3, 0)	(0, 0, 1), (1, 3, 1), (2, 1, 1), (3, 4, 1), (4, 2, 1), (2, 1, 0)
(1, 3, 1)	(0, 3, 1), (1, 1, 1), (2, 4, 1), (3, 2, 1), (4, 0, 1), (2, 1, 0)
(1, 3, 2)	(0, 1, 1), (1, 4, 1), (2, 2, 1), (3, 0, 1), (4, 3, 1), (2, 1, 0)
(1, 3, 3)	(0, 4, 1), (1, 2, 1), (2, 0, 1), (3, 3, 1), (4, 1, 1), (2, 1, 0)
(1, 3, 4)	(0, 2, 1), (1, 0, 1), (2, 3, 1), (3, 1, 1), (4, 4, 1), (2, 1, 0)
(1, 4, 0)	(0, 0, 1), (1, 1, 1), (2, 2, 1), (3, 3, 1), (4, 4, 1), (1, 1, 0)
(1, 4, 1)	(0, 1, 1), (1, 2, 1), (2, 3, 1), (3, 4, 1), (4, 0, 1), (1, 1, 0)
(1, 4, 2)	(0, 2, 1), (1, 3, 1), (2, 4, 1), (3, 0, 1), (4, 1, 1), (1, 1, 0)
(1, 4, 3)	(0, 3, 1), (1, 4, 1), (2, 0, 1), (3, 1, 1), (4, 2, 1), (1, 1, 0)
(1, 4, 4)	(0, 4, 1), (1, 0, 1), (2, 1, 1), (3, 2, 1), (4, 3, 1), (1, 1, 0)
(0, 1, 0)	(0, 0, 1), (1, 0, 1), (2, 0, 1), (3, 0, 1), (4, 0, 1), (1, 0, 0)
(0, 1, 1)	(0, 4, 1), (1, 4, 1), (2, 4, 1), (3, 4, 1), (4, 4, 1), (1, 0, 0)
(0, 1, 2)	(0, 3, 1), (1, 3, 1), (2, 3, 1), (3, 3, 1), (4, 3, 1), (1, 0, 0)
(0, 1, 3)	(0, 2, 1), (1, 2, 1), (2, 2, 1), (3, 2, 1), (4, 2, 1), (1, 0, 0)
(0, 1, 4)	(0, 1, 1), (1, 1, 1), (2, 1, 1), (3, 1, 1), (4, 1, 1), (1, 0, 0)
(0, 0, 1)	(0, 1, 0), (4, 1, 0), (3, 1, 0), (2, 1, 0), (1, 1, 0), (1, 0, 0)

The other predistribution scheme involves generalized quadrangles. Three known designs for generalized quadrangles have been used: $GQ(q, q)$, $GQ(q, q^2)$, and $GQ(q^2, q^3)$. The construction of $GQ(q, q)$, $GQ(q, q^2)$, and $GQ(q^2, q^3)$ has been done from $PG(4, q)$, $PG(5, q)$, and $H(4, q^2)$, respectively. Details can be found in [12, Section B]. Although the first scheme using symmetric design results in full connectivity of the network, the resiliency is poorer than in the second scheme using generalized quadrangles.

24.6.2 Lee and Stinson's Schemes

Lee and Stinson [45] formalized the definition of key predistribution schemes using set systems. They introduced the concept of common intersection designs [46] and discussed the use of block graphs for sensors, since by this design every pair of nodes is connected by a maximum of two hop paths. They also used transversal designs for key predistribution. The following construction of a transversal design TD(k, r) [45] is used.

1. $X = \{(x, y) : 0 \le x < k, 0 \le y < r\}$
2. For all i, $G_i = \{(i, y) : 0 \le y < r\}$
3. $\mathcal{A} = \{A_{i,j} : 0 \le i < r \;\&\; 0 \le j < r\}$

Each of the r^2 nodes (r is a prime power) is assigned a set of k keys, in such a way that node (i, j) receives the keys $\{(x, xi + j \bmod r) : 0 \le x < k\}$.

The design is such that two nodes share 0 or 1 common key. The expected number of common keys between two nodes is given by $p_1 = \frac{k(r-1)}{r^2-1} = \frac{k}{r+1}$. When two nodes want to communicate they broadcast their identifiers (i_1, j_1) and (i_2, j_2). The nodes can communicate if they share some common key. This happens when $xi_1 + j_1 = xi_2 + j_2 \pmod r$, such that
$x = (j_2 - j_1)(i_1 - i_2)^{-1} \pmod r < k$. Sect. 24.1.1 shows how shared keys are calculated in this scheme. A generalization of the above scheme was made by Lee and Stinson [47]. A TD(t, k, p) (where p is prime and $t \le k \le p$) is constructed in the following way.

Let $X_1 \subseteq X$, $|X_1| = k$, $X = X_1 \times \mathbb{Z}_p$. For $x \in X_1$ define $H_x = \{x\} \times \mathbb{Z}_p$ and define $\mathcal{H} = \{H_x : x \in X_1\}$. For every ordered t-tuple $\mathbf{c} = (c_0, c_1, \ldots, c_{t-1}) \in (\mathbb{Z}_p)^t$, define a block $A_c = \left\{\left(x, \sum_{i=0}^{t-1} c_i x^i\right) : x \in X_1\right\}$ and $\mathcal{A} = \{A_c : \mathbf{c} \in (\mathbb{Z}_p)^t\}$. Here $\mu_2(A, A')$ between two blocks A and A' is defined as $\mu_2(A, A') = |\{A'' : |A \bigcap A''| = |A' \bigcap A''| = 2\}|$, where $\mu_2 = \min\{\mu_2(A, A') : |A \bigcap A'| \le 1\}$ and η denotes the number of nodes in the intersection of the neighborhood of the two nodes U_i and U_j within communication range.

A_c are the blocks of the design. We can see that for $t = 2$ the construction is same as given in [45]. In [44] a quadratic scheme with TD$(3, k, p)$ is considered.

Let p be a prime and let F_p be a finite field of order p. We define a set system $(X, \mathcal{A})$ such that $X = D \times F_p$, $D = \{0, \ldots, k-1\}$ and $\mathcal{A} = \{A_{a,b,c} : a, b, c \in F_p\}$, where $A_{a,b,c} = \{(x, ax^2 + bx + c) : x \in D\}$.

The $s - connectivity$ of such a network is given by $Pr_1 = \frac{k(k-1)}{2(p^2+p+1)}$. The probability that two nodes are connected by paths of length 2 is given by $Pr_2 = \left(1 - \frac{k(k-1)}{2(p^2+p+1)}\right)\left(1 - \left(1 - \frac{\mu_2}{p^3-2}\right)^{\eta}\right)$. Probability that a link is broken when s nodes are compromised is fail$(s) = 1 - 2\left(1 - \frac{p^2-2}{p^3-2}\right)^s + \left(1 - \frac{2p^2-p-2}{p^3-2}\right)^s$.

The resiliency of the linear scheme is better than the quadratic scheme. A multiple space scheme has also been presented in [47].

Another grid-based deployment scheme was proposed by Chakrabarti [13]. This was further studied by Ruj et al. [78]. Here the predistribution scheme uses transversal design. The keys are distributed following the transversal design as done by Lee and Stinson. Blackburn et al. [5] proposed a grid-based deployment schemes in which the keys are predistributed using Costas arrays and distinct difference configuration.

24.6.3 Chakrabarti–Maitra–Roy Scheme

Chakrabarti et al. [14, 15] proposed a hybrid key predistribution scheme by merging the blocks in combinatorial designs. They considered the blocks constructed from the transversal design proposed by Lee and Stinson and randomly selected them and merged them to form the sensor nodes. Though this scheme increases the number of the keys per node, it improves the resiliency of the network. The probability that two nodes share a common key is also higher. Thus it has a better connectivity. Consider a transversal design TD(r, k). Let $v = rk$, $b = r^2$, and z blocks be selected independently at random to form a sensor node. We note the following features about the design.

1. There will be $N = \left\lfloor \frac{b}{z} \right\rfloor$ sensor nodes.
2. The probability that any two nodes share no common key is $(1 - p_1)^{z^2}$, where $p_1 = \frac{k}{r+1}$.
3. The expected number of keys shared between two nodes is $z^2 p_1$.
4. Each node will contain M distinct keys, where $zk - \binom{z}{2} \leq M \leq zk$. The average value of M is $\hat{a} = zk - \binom{z}{2}\frac{k}{r+1}$.
5. The expected number of links in the merged system is $\hat{L} = \left(\binom{r^2}{2} - \binom{z}{2}\left\lfloor \frac{r^2}{z} \right\rfloor\right)$ $\frac{k}{r+1} - (r^2 \mod z)k$.
6. Each key will be present in Q nodes, where $\left\lceil \frac{r}{2} \right\rceil \leq Q \leq r$. The average value of Q is $\hat{Q} = \frac{1}{kr}\left(\left\lfloor \frac{b}{z} \right\rfloor\right)\left(zk - \binom{z}{2}\frac{k}{r+1}\right)$.
7. Given that s nodes are compromised the fraction of links broken is given by

$$\text{Fail}(s) = \frac{\sum_{i=1}^{z^2} \frac{\binom{\gamma}{i}}{\binom{kr}{i}} \binom{z^2}{i} \left(\frac{k}{r+1}\right)^i \left(1 - \frac{k}{r+1}\right)^{z^2-i}}{1 - \left(1 - \frac{k}{r+1}\right)^2} \text{ where } \gamma = szk\left(1 - \frac{sz-1}{2(r+1)}\right)$$

In this scheme each of the nodes has the identifiers of the blocks which form that node. These identifiers are stored as a list in the node. To find a shared common key between two blocks, say U_1 and U_2, U_1 compares each identifier with z identifiers of U_2. For this, $O(z \log N)$ bits must be broadcasted by each node and z^2 inverse calculations are done.

24.6.4 *Ruj and Roy Scheme*

A key predistribution scheme using partially balanced incomplete block designs (PBIBD) was proposed by Ruj and Roy [80]. We defined a BIBD(v, b, r, k, λ) design earlier in this section. A pair of elements occur in exactly λ elements. A m-associate PBIBD$(v, b, r, k, \lambda_1, \lambda_2, \dots, \lambda_m)$ is a design which consists of v elements. There are b blocks each of size k, such that each element is repeated in r blocks. The difference from BIBD is that a pair of elements together occur either in λ_1 or in $\lambda_2, \dots, \lambda_m$ blocks. A PBIBD is symmetric, if $v = b$.

The authors [80] use a two-associate PBIBD. The underlying construction comes from a triangular PBIBD design [21]. The PBIBD is constructed in the following way. Consider the matrix A given by

$$A = \begin{array}{cccccc} * & 1 & 2 & \cdots & n-2 & n-1 \\ 1 & * & n & n+1 & \cdots & 2n-3 \\ 2 & n & * & 2n-2 & \cdots & 3n-6 \\ \vdots & \vdots & \vdots & * & \vdots & \vdots \\ n-1 & 2n-3 & \cdots & \cdots & n(n-1)/2 & * \end{array}$$

We know that the matrix is symmetric about the principal diagonal. Corresponding to each position (x, y) a block is created which consists of elements in the xth row and yth column. So there are $n(n-1)/2$ elements and $n(n-1)/2$ blocks. Each block contains $2(n-1)$ elements. A pair of elements occur together either in four or $n-2$ blocks. Any two blocks have either four or $n-2$ elements in common. So $v = b = n(n-1)/2$, $r = k = 2(n-2)$, $\lambda_1 = 4$, and $\lambda_2 = n-2$. It can be noted that given the position (x, y) of the matrix A, the element $a_{x,y} = a_{y,x}$ in the xth row and yth column is given by

$$a_{x,y} = \begin{cases} *, \text{ for } x = y \\ y - x, \text{ for } x = 1, x < y \\ x - y, \text{ for } y = 1, x > y \\ n + y - x - 1, \text{ for } x = 2, x < y \\ n + x - y - 1, \text{ for } y = 2, x > y \\ (x-1)n - (x+1)(x-2)/2 + (y - x - 1), \text{ for } x < y, x > 2 \\ (y-1)n - (y+1)(y-2)/2 + (x - y - 1). \text{ for } x > y, y > 2 \end{cases}$$

Each entry in the matrix corresponds to a node. Each node $n_{a_{i,j}}$, $(i \neq j)$ in position (i, j) of the matrix has the key identifiers $\{a_{x,j} : x = 1, 2, \ldots, n$, and $x \neq i, j\} \bigcup \{a_{i,y} : y = 1, 2, \ldots, n$ and $y \neq i, j\}$. Thus there are $n(n-1)/2$ sensor nodes, each having $2(n-2)$ keys. Any two nodes share either $n-2$ (or 4) keys depending on the fact that are (or are not) in the same row or same column.

Example 11 We present an example to demonstrate the above scheme. We consider $n = 5$. Then the array A containing the elements will be represented by

$$A = \begin{array}{ccccc} * & 1 & 2 & 3 & 4 \\ 1 & * & 5 & 6 & 7 \\ 2 & 5 & * & 8 & 9 \\ 3 & 6 & 8 & * & 10 \\ 4 & 7 & 9 & 10 & * \end{array}$$

There are 10 sensor nodes $n_i, i = 1, 2, \ldots, 10$, and 10 keys with identifiers $a_{i,j} (i \neq j)$ in the key pool. Each node has $2(n-2) = 6$ keys. For example, node $n_{a_{2,4}} = n_6$ will contain the keys having identifiers 3, 8, 10 (belonging to the same column as 6) and 1, 5, 7 (belonging to the same row as 6). The sensor nodes $n_i, i = 1, 2, \ldots, 10$, have the following keys: $n_1 : \{2, 3, 4, 5, 6, 7\}$, $n_2 : \{1, 3, 4, 5, 8, 9\}$, $n_3 : \{1, 2, 4, 6, 8, 10\}$, $n_4 : \{1, 2, 3, 7, 9, 10\}$, $n_5 : \{1, 2, 6, 7, 8, 9\}$, $n_6 : \{1, 3, 5, 7, 8, 10\}$, $n_7 : \{1, 4, 5, 6, 9, 10\}$, $n_8 : \{2, 3, 5, 6, 9, 10\}$, $n_9 : \{2, 4, 5, 7, 8, 10\}$, $n_{10} : \{3, 4, 6, 7, 8, 9\}$.

Let nodes $n_i = n_{a_{x,y}}$ and $n_j = n_{a_{x',y'}}$ want to communicate with each other. For this purpose the location (x, y) of the node in the array A is stored in the node. Nodes broadcast their position in the array A.

Given any node n_i at location (x, y), it can find the identifiers of the keys in common with another node n_j at position (x', y') in the following way.

1. If $x = x'$, then $a_{x,t}$ and $a_{y,y'}$ are the common keys between n_i and n_j for $t = 1, 2, \ldots, n$ and $t \neq x, y, y'$.
2. If $y = y'$, then $a_{t,y}$ and $a_{x,x'}$ are the common keys between n_i and n_j for $t = 1, 2, \ldots, n$ and $t \neq x, y, x'$.
3. If $x \neq x'$ and $y \neq y'$, then the keys $a_{x,x'}$, $a_{x,y'}$, $a_{y,x'}$ and $a_{y,y'}$ are common between n_i and n_j.

Since there are more than one key in common, the nodes can choose any of the common keys for communication. Since $a_{x,y}$ can be calculated in constant time, key agreement can be done in $O(1)$ time. Also the memory overhead is $O(\log\ n) = O\left(\log \sqrt{N}\right)$ bits, since only the position of the node in the array is sent.

The construction above can be used to construct a key predistribution scheme having double the number of sensor nodes as the previous scheme. In addition to the matrix A another matrix A' is constructed in the following way.

$$A' = \begin{array}{cccccc} * & 1 & n & \cdots n(n-1)/2-2 & & n(n-1)/2 \\ 1 & * & 2 & \cdots & \cdots & n(n-1)/2-1 \\ n & 2 & * & 3 & \cdots & n(n-1)/2-3 \\ \vdots & \vdots & \vdots & * & \vdots & \vdots \\ n(n-1)/2 & n(n-3)/2-1 & \cdots & \cdots & n-1 & * \end{array}$$

The first $n(n-1)/2$ nodes are each assigned $2(n-2)$ according to the matrix A. The next $n(n-1)/2$ nodes are each assigned $2(n-2)$ elements according to the matrix A'. Suppose the element in the (i, j)th position of the matrix A' is $a'(i, j)$. The node in position $(i, j)(i \neq j)$ of the matrix A' is denoted by $n_{n(n-1)/2+a'(i,j)}$ and has the keys with identifiers $\{a'_{x,j} : x = 1, 2, \dots, n, \text{ and } x \neq i, j\} \bigcup \{a'_{i,y} : y = 1, 2, \dots, n \text{ and } y \neq i, j\}$.

Example 11 (contd). We consider the following matrix.

$$A' = \begin{array}{ccccc} * & 1 & 5 & 8 & 10 \\ 1 & * & 2 & 6 & 9 \\ 5 & 2 & * & 3 & 7 \\ 8 & 6 & 3 & * & 4 \\ 10 & 9 & 7 & 4 & * \end{array}$$

The nodes 11–20 are assigned keys in the following way : $n_{10+a'_{1,2}} = n_{11} : \{2, 5, 6, 9, 8, 10\}$, $n_{10+a'_{2,3}} = n_{12} : \{1, 3, 5, 6, 7, 9\}$, $n_{10+a'_{3,4}} = n_{13} : \{2, 4, 5, 6, 7, 8\}$, $n_{10+a'_{4,5}} = n_{14} : \{3, 6, 7, 8, 9, 10\}$, $n_{10+a'_{1,3}} = n_{15} : \{1, 2, 3, 7, 8, 10\}$, $n_{10+a'_{2,4}} = n_{16} : \{1, 2, 3, 4, 8, 9\}$, $n_{10+a'_{3,5}} = n_{17} : \{2, 3, 4, 5, 9, 10\}$, $n_{10+a'_{1,4}} = n_{18} : \{1, 3, 4, 5, 6, 10\}$, $n_{10+a'_{2,5}} = n_{19} : \{1, 2, 4, 6, 7, 10\}$, and $n_{10+a'_{1,5}} = n_{20} : \{1, 4, 5, 7, 8, 9\}$. So now the entire network has 20 nodes, each having six keys as before. Also the key pool remains the same. So if the first 10 nodes have already been deployed, another set of 10 nodes can be deployed without redistributing keys in the old nodes.

This scheme has several advantages. All the nodes can directly communicate with each other, thus reducing the time for communication. The resiliency is improved compared to the other approaches. The system can also be scaled to twice the original size. The number of keys is $O\left(\sqrt{N}\right)$, where N is the size of the network.

The drawback of this scheme is that since some nodes can share a large number of keys, resiliency drops rapidly when large number of nodes are compromised.

24.6.5 Key Predistribution Schemes Using Codes

In [3], Al-Shurman and Yoo proposed a key management scheme based on maximum distance separable (MDS) codes to satisfy the properties of cover free family (CFF). A set system $(X, \mathcal{A})$ is called a t cover-free family (or $t-$ CFF) if, for any t blocks $A_1, A_2, \ldots, A_t \in \mathcal{A}$ and any other block $A_0 \in \mathcal{A}$, we have

$$A_0 \subsetneq \bigcup_{i=1}^{t} A_i$$

Their construction satisfies the properties of CFF with certain probability. CFF was first studied by Kautz [43]. CFF has also been studied in [89, 90]. Chan [16] proposed a practical scheme, based on probabilistic method called distributed key selection to construct CFF.

The main idea was to use MDS code to generate node key chains. To construct the public matrix G of Blom's scheme (refer to Sect. 24.2.1), Reed–Solomon codes are used. G is defined in GF(q) as k rows and n columns with $d = n - k + 1$. The ijth element of G is given by $G_{ij} = \alpha^{(i-1)(j-1)}$ where α is the primitive element of q. Node i generates a random vector v_i of length k and calculates k_i, the key chain. From the definition of CFF, we see that even if t nodes are compromised, not all keys in an uncompromised node are exposed. Al-Shurman et al's scheme satisfies the CFF property with certain probability.

In [82], Ruj and Roy proposed a key predistribution scheme using codes. This is a generalized scheme which can be used with any code with suitable parameters. The basic approach is to generate key chains from a key pool in a deterministic fashion. Let $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ be a codeword. The codewords are mapped to sensor nodes. For any sensor U_i the codeword $\mathbf{c}$ is used such that the keys $\{(c_i, i) : i = 1, 2, \ldots, n\}$ are in its key ring. Suppose (n, M, d, q)-code [87, Section 10.2] having length n, distance d, and number of codewords equal to M. Let Q be the set of symbols, such that $|Q| = q$. The key pool consists of keys $\{(i, j) : i \in Q, j = 1, 2, \ldots, n\}$. The size of the key pool is qn. Suppose the ith codeword be $\left(a_1^{(i)}, a_2^{(i)}, \ldots, a_n^{(i)}\right)$. We assign the key identifiers $\left(a_j^{(i)}, j\right)$ for $j = 1, 2, \ldots, n$ to sensor U_i.

Thus each sensor contains n keys. Suppose d be the minimum distance between two codewords x and y, then the number of common keys between two sensor nodes x and y is at the maximum $n - d$. The authors use Reed–Solomon codes [74], but in general any code with a suitable parameter can be considered.

Let GF(q) be a finite field of $q > 2$ elements. Let $\mathcal{P}$ be the set of polynomials over GF(q) of degree at most $k-1$. $|\mathcal{P}| = q^k$. $d = n-k+1$. Let $F_q^* = \{\alpha_1, \alpha_2, \ldots, \alpha_{q-1}\}$ be the set of non-zero elements of GF(q). For each polynomial $p_i(x) \in \mathcal{P}$, we define

$c_{p_i} = (p_i(\alpha_1), p_i(\alpha_2), \ldots, p_i(\alpha_{q-1}))$ to be the ith codeword of length $q - 1$. We define $\mathcal{C} = \{c_{p_i} : p_i(x) \in \mathcal{P}\}$. Thus $\mathcal{C}$ is a Reed–Solomon code.

Example 12 Let us consider a Reed–Solomon code having parameters $q = 4, n = 3, k = 2$. Thus $d = n - k + 1 = 2$. Let the elements of the field be represented by $\{0, 1, \beta = 2, 1 + \beta = 3\}$. Let the codewords be as shown below.

$$\begin{pmatrix} 0&0&0\\ 1&1&1\\ 2&2&2\\ 3&3&3\\ 1&2&3\\ 0&3&2\\ 3&0&1\\ 2&1&0\\ 2&3&1\\ 3&2&0\\ 0&1&3\\ 1&0&2\\ 3&1&2\\ 2&0&3\\ 1&3&0\\ 0&2&1 \end{pmatrix}$$

This results in a network having $q^2 = 16$ nodes, each having three keys. The node ids, polynomials corresponding to the nodes, and keys belonging to the nodes are given in Table 24.4. For example, node 5 corresponds to the polynomial $p_5(x) =$

Table 24.4 Nodes, corresponding polynomials, and keys

Node id (i)	Polynomial ($p_i(x)$)	Keys
0	0	$\{(0,1),(0,2),(0,3)\}$
1	1	$\{(1,1),(1,2),(1,3)\}$
2	2	$\{(2,1),(2,2),(2,3)\}$
3	3	$\{(3,1),(3,2),(3,3)\}$
4	x	$\{(1,1),(2,2),(3,3)\}$
5	$1+x$	$\{(0,1),(3,2),(2,3)\}$
6	$2+x$	$\{(3,1),(0,2),(1,3)\}$
7	$3+x$	$\{(2,1),(1,2),(0,3)\}$
8	$2x$	$\{(2,1),(3,2),(1,3)\}$
9	$1+2x$	$\{(3,1),(2,2),(0,3)\}$
10	$2+2x$	$\{(0,1),(1,2),(3,3)\}$
11	$3+2x$	$\{(1,1),(0,2),(2,3)\}$
12	$3x$	$\{(3,1),(1,2),(2,3)\}$
13	$1+3x$	$\{(2,1),(0,2),(3,3)\}$
14	$2+3x$	$\{(1,1),(3,2),(0,3)\}$
15	$3+3x$	$\{(0,1),(2,2),(1,3)\}$

$1+x$. Therefore key identifiers contained in it are $(p_5(1), 1)$, $(p_5(\beta), 2)$, and $(p_5(1+\beta), 3)$, i.e., $(0, 1)$, $(\beta + 1, 2)$, and $(\beta, 3)$, i.e., $(0, 1)$, $(3, 2)$, $(2, 3)$.

The advantage of this scheme is that new nodes can be added without reloading the keys in the existing sensor nodes.

Dong et al. [24] design a key predistribution scheme based on three-designs. Wei and Wu [92] use construction from set systems which optimize the product construction of [27, 48]. The scheme is based on the product of key distribution scheme and set systems. They deduce conditions of the set systems that provide optimum connectivity and resiliency of the network.

24.7 Key Predistribution in Multi-hop Networks

In multi-hop networks nodes may not be within communication range of one another and so must communicate through intermediary nodes. The networks are modeled as *unit disk graphs*. Radio (transmit and sense) coverage area for each sensor node is assumed to be a circle with radius r (radio range) centered at its (x, y)-coordinate. If two nodes are within each other's radio range, then there is a link between them.

When considering the multi-hop networks the deployment pattern of the nodes become important, because the nodes in the physical proximity must be able to communicate efficiently. This deployment knowledge has been exploited by various researchers for improving the security of the systems. According to these schemes the sensors are deployed according to some given pattern. The pattern is then exploited in key predistribution.

The first key predistribution scheme for multi-hop networks was proposed by Liu and Ning [49]. They proposed two predistribution schemes both of which take advantage of the deployment knowledge of sensor nodes. The first scheme called the *closest pairwise scheme* was a modification of the pairwise key predistribution scheme. The second predistribution scheme uses the polynomial-based key predistribution scheme of Blundo et al. [7]. Independently Du et al. proposed a key predistribution scheme using deployment knowledge in [26], which they extended in [29]. This scheme uses a grid group-based deployment scheme in which sensors are deployed in groups, such that a group of sensors are deployed at a single deployment point, and the pdfs (probability distribution functions) of the final resident points of all the sensors in a group are the same. The key predistribution scheme uses multiple space Blom scheme as in [27, 30]. There are several other schemes available in literature, which employ deployment knowledge during key predistribution [5, 26, 29, 39, 40, 49, 85, 95, 96, 98].

In multi-hop networks, since nodes communicate via intermediary nodes, steps should be taken such that even if the intermediary nodes are compromised, the path-keys between the communicating nodes are not known. Sometimes heterogeneous networks are constructed. In such networks there are different types of nodes having varying battery power and memory. Some nodes are tamper proof and others are not. They also differ in the mode of operation and are placed in different environments.

In a heterogeneous/hierarchical network there are three types of nodes: *sensor nodes*, *cluster heads* (CH), and a *base station* (BS), also called *command node* in increasing order of battery and memory capacities. The BS is very powerful and is placed in a safe place and is thus assumed to be fully resilient to adversarial attacks. The CHs and sensors nodes are deployed in the adversarial region. The CHs are responsible for aggregating the information from the sensor nodes and forwarding to the BS. Thus the CHs are more powerful than the sensor nodes and fewer in number compared to the sensor nodes.

Some key predistribution schemes in heterogeneous networks have been proposed are SPINS protocol by Perrig et al. [71] and localized encryption and authentication protocol (LEAP) by Zhu et al. [99]. SPINS has two components: secure network encryption protocol (SNEP) and μTESLA (time efficient stream loss-tolerant authentication). SNEP provides data confidentiality and two-party data authentication. μTESLA provides efficient broadcast for severely resource-constrained environments. μTESLA is an improvement of TESLA proposed by Perrig et al. [70].

A number of cluster-based schemes [20, 22, 42, 69] have been proposed in which sensor nodes are divided into groups or clusters, each having one or more cluster heads. The sensor nodes in each cluster send their data to its cluster head(s) (CH), which processes it and sends it to the BS. Another cluster-based approach in heterogeneous networks was taken by Oliveira et al. [66, 67]. The scheme is known as SecLEACH, which is a modification of the low energy adaptive clustering hierarchy—LEACH [38] (proposed by Heinzelman et al.) and F-LEACH [34](proposed Ferreira et al.). Other heterogeneous schemes include [31, 54]. Heterogeneous schemes using combinatorial designs have been proposed by Ruj and Roy [81] and Younis et al. [95].

24.8 Conclusion

Having studied different kinds of predistribution schemes, probabilistic, deterministic, and hybrid, we are now in a position to compare them. In Table 24.5, we compare the scalability, key connectivity, resiliency, key storage, communication overhead, and computation required for key establishment. We also state the nature of the key predistribution scheme—probabilistic, deterministic, and hybrid. To compute the resiliency we consider either of the two aspects.

1. Probability that a link is broken when a node is compromised.
2. We also consider t-secure resiliency meaning that the network remains connected if t or less nodes are compromised.

For larger expressions we refer to the respective section in the chapters where they appear.

To compute the key storage, either we give the exact number of keys or, when comparing the number of keys with the size of the network, we give the order in

Table 24.5 A table that compares the different key predistribution schemes

Scheme	Type	Scalability	Key connectivity	Resilience	Key storage	Overhead	Computation
Blom	Probabilistic	Not Scalable	1	t-secure	$t+1$	$t+1$	$t+1$
Blundo	Probabilistic	Scalable	1	t-secure	$(t+1)\log q$	$O(\log N)$	$t+1$
ES	Probabilistic	Scalable	$\frac{((\lvert X\rvert-k)!)^2}{(\lvert X\rvert-2k)!\lvert X\rvert!}$	$k/\lvert X\rvert$	k	$O(k\log\lvert X\rvert)$	$k\log\lvert k\rvert$
Q-Composite	Probabilistic	Limited scalability	Given in [19, Section 5.2]	$\binom{k}{q}$	k	$O(k\log\lvert X\rvert)$	$k\log\lvert k\rvert$
Random pair-wise schemes							
Chan–Perrig–Song [19]	Probabilistic	Scalable	$\frac{((\lvert X\rvert-k)!)^2}{(\lvert X\rvert-2k)!\lvert X\rvert!}$	$k/\lvert X\rvert$	Np_c	$O(k\log\lvert X\rvert)$	$k\log\lvert k\rvert$
Liu–Ning–Li [48, 53]	Hybrid	Scalable	Given in [48, Section 4.1]	t-secure	$s'(t+1)\log q$	$s'\log\lvert\mathcal{F}\rvert$	$t+1$
Zhu et al. [100]	Probabilistic	Scalable	1	Given in [100, Section 4.1]	k	$O(\log N)$	n, n number of shares
Grid-based Schemes							
PIKE [18]	Deterministic	Not scalable	$1/\sqrt{N}$	$1/\sqrt{N}$	$O(\sqrt{N})$	$O(\log N)$	$O(1)$
Liu–Ning–Du [52](hash-key)	Deterministic	Not scalable	Given in [52, Section 5.2]	Given in [52, Section 5.4]	$(m+n)/2$	$O(\log N)$	$O(H)$, H is the time taken to compute hash functions

Table 24.5 (continued)

Liu–Ning–Du [52](polynomial)	Hybrid	Scalable	Given in [52, Section 5.4]	Given in [52, Section 5.4]	$\frac{(m+n)/2}{t}\log q$	$O(t+1)$	$O(t+1)$
Martin–Paterson–Stinson [61]	Deterministic	Not scalable	k/n if in the same group $\frac{n-1}{n^2/\lambda-1}+\frac{n^2/\lambda-n}{n^2/\lambda-1}mn$ if in the different group	Given in [61]	$(m+1)(t+1)$	$O(\log N)$	$O(1)$
Combinatorial design-based schemes							
Çamtepe and Yener [10, 12] (symmetric)	Deterministic	Not scalable	1	$\frac{q-1}{q^2+q+1}$	$O(\sqrt{N})$	$O(\log N)$	$O(1)$
Lee–Stinson [45]	Deterministic	Not scalable	$\frac{k}{r+1}$	$\frac{r-2}{b-2}$	k	$O(\log N)$	$O(1)$
Chakrabarti et al. [14, 15]	Hybrid	Not scalable	Given in [14, 15]	Given in [14, 15]	$zk-\binom{z}{2}\frac{k}{r+1}$	$O(z\log N)$	$z\log\lvert X\rvert$
Ruj and Roy [80]	Deterministic	Scalable	1	Given in [80, Section 4]	$O\left(\sqrt{N}\right)$	$O(\log N)$	$O(1)$

terms of the size of the network. In some cases we simply write the number of keys as k meaning that the number of keys can be chosen arbitrarily, that there is no relation with the parameters of the design. The same holds for the communication overhead, where we either give the exact number of bits required or its order. The computation required for shared-key discovery is calculated in the same way. The symbols have their usual meaning as discussed in the chapter.

We observe that deterministic designs have efficient (both in terms of communication and computation) shared-key discovery algorithms because of the inherent patterns they possess. However, as Martin [58] pointed out that the deterministic schemes may not be flexible enough to trade-off the important parameters. For example, we see that the predistribution using combinatorial structures have nice shared-key discovery but have some scalability, storage, and resiliency issues. On the other hand, the random schemes are highly configurable.

We also note that there is a tradeoff between connectivity and resiliency. As the connectivity increases, resiliency increases up to a certain point. However, as more nodes get compromised the resiliency decreases rapidly as connectivity increases. Also we note that full connectivity (connectivity between every pair of nodes) may not be required. It is sufficient if the nodes within communication range are directly connected to each other. However, Pietro et al. [73] proved by mathematical analysis that connectivity via secure links and resiliency against malicious attacks can be obtained simultaneously. They show that there is a discrepancy between the otherwise widely used Erdös–Rényi model and the real networks generated by random predistribution schemes. They defined a *redoubtable* network as one in which an adversary has to compromise a large number of nodes to compromise the confidentiality of the network. More formally, a network is redoubtable if the probability that a collusion of $o(N)$ nodes uniformly chosen at random in the network compromises a constant fraction of the network links is $o(1)$. Suppose the aim of the adversary is to split the network into two large chunks, both of linear size to compromise all links between them and thus partition the network. They show that the probability that this is achieved by compromising a sublinear number of nodes is negligible. Assume that after compromising $o(N)$ nodes, there exists two disjoint set of vertices A and B such that all edges between A and B are compromised and both A and B have linear size. Such a pair A, B exists then the network is said to have a bad split. A network is *unsplittable* if there are no bad splits after the adversary has compromised $o(N)$ networks. They show that if $v \geq N \log N$ and the relation

$$\frac{k^2}{v} \sim \frac{\log N}{N}$$

holds, then the networks are not only connected with a high probability but also redoubtable. They also show that the network is unsplittable if the above conditions hold.

Though a lot of research is being carried out, the following problems deserve to be explored further. The path-key establishment poses a threat to security because if the intermediary nodes are compromised then the path is compromised. This is taken care of by multipath key establishment. However, it may be expensive finding all the paths. So a better method needs to be designed. Another problem is of efficient authentication techniques. Many of the authentication techniques make use of public key cryptography which may not be efficient for sensor networks. The implementation of the schemes needs to be addressed.

References

1. Tinyos. Available at http://www.tinyos.net. Accessed on November 28, 2009.
2. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, 2002.
3. M. Al-Shurman and S.-M. Yoo. Key pre-distribution using mds codes in mobile ad hoc networks. In: *ITNG*, IEEE Computer Society, pages 566–567, 2006.
4. S. Blackburn and S. Gerke. Connectivity of the uniform random intersection graph. *Discrete Mathematics*, 309:5130–5140, 2009.
5. S.R. Blackburn, T. Etzion, K. M. Martin, and M. B. Paterson. Efficient key predistribution for grid-based wireless sensor networks. In Safavi-Naini R., editor. ICITS, volume 5155 of *Lecture Notes in Computer Science*, Springer,Berlin, pages 54–69, 2008.
6. R. Blom. An optimal class of symmetric key generation systems. In: *Proceeding of EUROCRYPT*, *Lecture Notes in Computer Science*, Springer, Berlin, pages 335–338, 1984.
7. C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In: *Advances in Cryptology*: *Proceedings CRYPTO'92*, Lecture Notes in Computer Science, Santa Barbara, CA, 740:471–486, 1993.
8. R. Canetti, J. A. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOM*, pages 708–716, 1999.
9. D.W. Carman, P.S. Kruus, and B.J. Matt. Constrains and approaches for distributed sensor network security, Technical report 00-010. NAI Labs, Glenwood, MD, 2000.
10. S.A. Camtepe, and B. Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. In Samarati P., Ryan P. Y. A, Gollmann D., and Molva R., editors. ESORICS, volume 3193 of *Lecture Notes in Computer Science*, Springer, Berlin, pages 293–308, 2004.
11. S. A. Camtepe and B. Yener. Key distribution mechanisms for wiless sensor networks: A survey,. Technical Report TR-05-07 Rensselaer Polytechnic Institute, Computer Science Department, March 2005.
12. S.A.Camtepe and B. Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. IEEE/ACM Transport. Network, 15(2):346–358, 2007.
13. D. Chakrabarti. Applications of combinatorial designs in key pre-distribution in sensor networks. Ph.D. Thesis, Indian Statistical Institute, India, September 2007.
14. D. Chakrabarti, S. Maitra, and B. Roy. A key pre-distribution scheme for wireless sensor networks: Merging blocks in combinatorial design. In Zhou J., Lopez J., Deng R. H, Bao F., editors. ISC, volume 3650 of *Lecture Notes in Computer Science*, Springer, Berlin, pages 89–103, 2005.
15. D. Chakrabarti, S. Maitra, and B. Roy. A key pre-distribution scheme for wireless sensor networks: Merging blocks in combinatorial design. *International Journal of Infermation Security*, 5(2):105–114, 2006.
16. A.C.-F. Chan. Distributed symmetric key management for mobile ad hoc networks. In: INFOCOM, 2004.
17. H. Chan, V.D. Gligor, A. Perrig, and G. Muralidharan. On the distribution and revocation of cryptographic keys in sensor networks. *IEEE Transactions on Dependable and Secure Computing*, 2(3):233–247, 2005.

18. H. Chan and A. Perrig. PIKE: Peer intermediaries for key establishment in sensor networks. In: *INFOCOM*, IEEE, pages 524–535, 2005.
19. H. Chan, A. Perrig, and D. X. Song. Random key predistribution schemes for sensor networks. In: *IEEE Symposium on Security and Privacy*, IEEE Computer Society, pages 197–213, 2003.
20. Y. Cheng and D.P. Agrawal. An improved key distribution mechanism for large-scale hierarchical wireless sensor networks. *Ad Hoc Networks*, 5(1):35–48, 2007.
21. W. H. Clatworthy. Tables of Two-Associate-Class Partially Balanced Designs. *NBS Applied Mathematics Series*, 63, 1973.
22. A. K. Das and I. Sengupta. An effective group-based key establishment scheme for large-scale wireless sensor networks using bivariate polynomials. In: *COMSWARE*, IEEE, pages 9–16, 2008.
23. T. Dierks and C. Allen. The TLS Protocol Version 1.0, 1999. RFC 2246, January 1999.
24. J. Dong, D. Pei, and X. Wang. A key predistribution scheme based on 3-designs. In Pei D., Yung M., Lin D., Wu C., editors. Inscrypt, volume 4990 of *Lecture Notes in Computer Science*, pages 81–92, Springer, Berlin, 2007.
25. J. R. Douceur. The Sybil Attack. In Druschel P., Kaashoek M. F., and Rowstron A. I. T., editors, IPTPS, volume 2429 of *Lecture Notes in Computer Science*, Springer, Berlin, pages 251–260, 2002.
26. W. Du, J. Deng, Y. S. Han, S. Chen, and P.K. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In: *INFOCOM*, pages 261–268, 2004.
27. W. Du, J. Deng, Y.S. Han, and P.K. Varshney. A pairwise key predistribution scheme for wireless sensor networks. In Jajodia S., Atluri V., and Jaeger T., editors. *ACM Conference on Computer and Communications Security*, ACM, pages 42–51, 2003.
28. W. Du, J. Deng, Y.S. Han, P.K. Varshney. A witness-based approach for data fusion assurance in wireless sensor networks. In: *Proceedings*. IEEE Global Telecommunications Conference. (GLOBECOM '03), 2003.
29. W. Du, J. Deng, Y.S. Han, and P.K. Varshney. A key predistribution scheme for sensor networks using deployment knowledge. *IEEE Transactions on Dependable and Secure Computing*, 3(1):62–77, 2006.
30. W. Du, J. Deng, Y.S. Han, P.K. Varshney, J. Katz, and A. Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Transactions on Information and System Security*, 8(2):228–258, 2005.
31. X. Du, Y. Xiao, M. Guizani, and H.-H. Chen. An effective key management scheme for heterogeneous sensor networks. *Ad Hoc Networks*, 5(1):24–34, 2007.
32. P. Erdös and A. Rényi. On random graphs. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.
33. L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In Atluri V., editor. *ACM Conference on Computer and Communications Security*, ACM, pages 41–47, 2002.
34. A. C. Ferreira, M. A. Vilaca, L. B. Oliveira, E. Habib, H. C. Wong, and A. A. F. Loureiro. On the security of cluster-based communication protocols for wireless sensor networks. In Lorenz P. and Dini R., editors. ICN (1), volume 3420 of *Lecture Notes in Computer Science*, Springer, Berlin, pages 449–458, 2005.
35. S. Ganeriwal and M. B. Srivastava. Reputation-based framework for high integrity sensor networks. In Setia S. and Swarup V., editors. *SASN*, ACM, pages 66–77, 2004.
36. G. Gaubatz, J.-P. Kaps, and B. Sunar. Public key cryptography in sensor networks–revisited. In Castelluccia C., Hartenstein H., Paar C., Westhoff D., editors. ESAS, volume 3313 of *Lecture Notes in Computer Science*, Springer, Berlin, pages 2–18, 2004.
37. N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing elliptic curve cryptography and RSA on 8-bit cpus. In Joye M. and Quisquater J.-J., editors. CHES, volume 3156 of *Lecture Notes in Computer Science*, Springer, Berlin, pages 119–132, 2004.
38. W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *IEEE Hawaii International Conference on System Sciences*, pages 1–10, 2000.

39. D. Huang and D. Medhi. Secure pairwise key establishment in large-scale sensor networks: An area partitioning and multigroup key predistribution approach. *TOSN*, 3(3):16:1–16:34, 2007.
40. D. Huang, M. Mehta, D. Medhi, and L. Harn. Location-aware key management scheme for wireless sensor networks. In: *2nd ACM workshop on Security of ad hoc and sensor networks, SASN, Washington, DC*, pages 29–42, 2004.
41. J. Hwang and Y. Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In Setia S., Swarup V., editors. *SASN*, ACM, pages 43–52, 2004.
42. G. Jolly, M. C. Kusçu, P. Kokate, and M. F. Younis. A low-energy key management protocol for wireless sensor networks. In ISCC, *IEEE Computer Society*, pages 335–340, 2003.
43. W.H. Kautz and R. C. Singleton. Nonrandom binary superimposed codes. *IEEE transaction on Information Theory*, 10:363–377, 1964.
44. J. Lee. Combinatorial approaches to key predistribution for Distributed Sensor Networks, 2005. Ph.D. Thesis, University of Waterloo, Waterloo, ON, 2005.
45. J. Lee and D. R. Stinson. A combinatorial approach to key predistribution for distributed sensor networks. In: *IEEE Wireless Communications and Networking Conference, WCNC 2005*, New Orleans, LA, pages 1200–1205, 2005.
46. J. Lee and D. R. Stinson. Common intersection designs. *Journal of Combinatorial Designs*, 14:251–269, 2006.
47. J. Lee and D. R. Stinson. On the construction of practical key predistribution schemes for distributed sensor networks using combinatorial designs. *ACM Transactions of Information and System Security*, 11(2):5:1–5:35, 2008.
48. D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In Jajodia S., Atluri V., Jaeger T. editors. *ACM Conference on Computer and Communications Security*, ACM, pages 52–61, 2003.
49. D. Liu and P. Ning. Location-based pairwise key establishments for static sensornetworks. In Setia S., Swarup V., editors. *SASN*, ACM, pages 72–82, 2003.
50. D. Liu, P. Ning, and W. Du. Detecting malicious beacon nodes for secure location discovery in wireless sensor networks. In ICDCS, *IEEE Computer Society*, pages 609–619, 2005.
51. D. Liu, P. Ning, and W. Du. Group-based key pre-distribution in wireless sensor networks. In Jakobsson M. and Poovendran R., editors. *Workshop on Wireless Security*, ACM, pages 11–20, 2005.
52. D. Liu, P. Ning, and W. Du. Group-based key predistribution for wireless sensor networks. *TOSN*, 4(2):11:1–11:30, 2008.
53. D. Liu, P. Ning, and R. Li. Establishing pairwise keys in distributed sensor networks. ACM Transactions on Information and System Security, 8(1):41–77, 2005.
54. B. Maala, Y. Challal, and A. Bouabdallah. HERO: Hierarchical key management protocol for heterogeneous wireless sensor networks. *Wireless Sensor and Actor Networks II*, 264:125–136, 2008.
55. F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error Correcting Codes*. North-Holland publishers, North-Holland, 1988.
56. D. Malan, M. Welsh, and M. Smith. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In: *First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON)*, pages 119–132, 2004.
57. D. J. Malan, M. Welsh, and M. D. Smith. Implementing public-key infrastructure for sensor networks. *TOSN*, 4(4):22:1–22:23, 2008.
58. Keith M. Martin. On the applicability of combinatorial designs to key predistribution for wireless sensor networks. In Chee Y.M., Li C., Ling S., Wang H., and Xing C., editors. IWCC, volume 5557 of *Lecture Notes in Computer Science*, Springer, Berlin, pages 124–145, 2009.
59. K. M. Martin and M. B. Paterson. An application-oriented framework for wireless sensor network key establishment. In: *Third Workshop on Cryptography for Ad-hoc Networks WCAN'07*, 2007.
60. K. M. Martin and M. B. Paterson. An application-oriented framework for wireless sensor network key establishment. *Electronic Notes in Theoretical Computer Science*, 192(2):31–41, 2008.

61. K. M. Martin, M. B. Paterson, and D. R. Stinson. Key predistribution for homogeneous wireless sensor networks with group deployment of nodes, 2008. Available at ePrint Cryptology archive 2008/412.
62. R. C. Merkle. Protocols for public key cryptosystems. In: *IEEE Symposium on Security and Privacy*, pages 122–134, 1980.
63. C. J. Mitchell, F. Piper. Key storage in secure networks. *Discrete Applied Mathematics*, 21:215–228, 1988.
64. A. Mohaisen, Y. J. Maeng, and D. H. Nyang. On grid-based key predistribution: Toward a better connectivity in wireless sensor network. In Washio T., Zhou Z.-Z, Huang J. Z, Hu X., Li J., Xie C., He J., Zou D., Li K.-C., and Freire M.M, editors. PAKDD Workshops, volume 4819 of *Lecture Notes in Computer Science*, Springer, Berlin, pages 527–537, 2007.
65. R. M. Needham. Dyption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.
66. L. B. Oliveira, A. C. Ferreira, M. A. Vilaça, H. C. Wong, M. l W. Bern, R. Dahab, and Antonio Alfredo Ferreira Loureiro. SecLEACH–On the security of clustered sensor networks. *Signal Processing*, 87(12):2882–2895, 2007.
67. L. B. Oliveira, H. C. Wong, M. W. Bern, R. Dahab, A. Alfredo, and F. Loureiro. SecLEACH–A random key distribution solution for securing clustered sensor networks. In: *NCA*, *IEEE Computer Society*, pages 145–154, 2006.
68. E. C. Park and I. F. Blake. Reducing communication overhead of key distribution schemes for wireless sensor networks. In: *ICCCN*, IEEE, pages 1345–1350, 2007.
69. M. B. Paterson, D. B. Stinson. Two attacks on a sensor network key distribution scheme of Cheng and Agrawal. *Journal of Mathematical Cryptology*, 2:393–403, 2008.
70. A. Perrig, R. Canetti, J. D. Tygar, and D. X. Song. Efficient authentication and signing of multicast streams over lossy channels. In: *IEEE Symposium on Security and Privacy*, pages 56–73, 2000.
71. A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar. SPINS: Security protocols for sensor netowrks. In: *MOBICOM*, pages 189–199, 2001.
72. R. D. Pietro, L. V. Mancini, and A. Mei. Energy efficient node-to-node authentication and communication confidentiality in wireless sensor networks. *Wireless Networks*, 12(6):709–721, 2006.
73. R. D. Pietro, L. V. Mancini, A. Mei, A. Panconesi, and J. Radhakrishnan. Redoubtable sensor networks. *ACM Transactions on. Information and System. Security*, 11(3):13:1–13:22, 2008.
74. I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics(SIAM)*, 8:300–304, 1960.
75. R. Roman, J. Zhou, and J. Lopez. On the security of wireless sensor networks. In Gervasi O., Gavrilova M. L, Kumar V., Laganàa., Lee H. P., Mun Y., Taniar D., and Tan C. J. K., editors. ICCSA (3), volume 3482 of *Lecture Notes in Computer Science*, Springer, Berlin, pages 681–690, 2005.
76. K. Römer and F. Mattern F. The design space of wireless sensor networks. *IEEE Wireless Communications*, 11(6):54–61, 2004.
77. S. Ruj. Application of combinatorial structures to key predistribution in sensor networks and traitor tracing. Ph.D. Thesis, Indian Statistical Institute, Kolkata, India, February, 2009.
78. S. Ruj, S. Maitra, and B. Roy. Key predistribution using transversal design on a grid of wireless sensor network. *Ad Hoc & Sensor Wireless Networks*, 5(3–4):247–264, 2008.
79. S. Ruj, A. Nayak, and I. Stojmenovic. Pairwise key predistribution using combinatorial designs. In preparation.
80. S. Ruj and B. Roy. Key predistribution using partially balanced designs in wireless sensor networks. In I. Stojmenovic, R. K. Thulasiram, L. T. Yang, W. Jia, M. Guo, and R. F.Mello, editors. ISPA, volume 4742 of *Lecture Notes in Computer Science*, Springer, Berlin, pages 431–445, 2007.
81. S. Ruj and B. Roy. Key predistribution using combinatorial designs for grid-group deployment scheme in wireless sensor networks. *ACM Transaction on Sensor Networks*, 6(1):4:1–4:28, 2009.

82. S. Ruj and B. K. Roy. Key predistribution schemes using codes in wireless sensor networks. In Yung M., Liu M., and Lin D., editors., Inscrypt, volume 5487 of *Lecture Notes in Computer Science*, Springer, Berlin, pages 275–288, 2008.
83. M. G. Sadi, D. S. Kim, and J. S. Park. GBR: Grid based random key predistribution for wireless sensor network. In: *ICPADS* (2), IEEE Computer Society, pages 310–315, 2005.
84. A. Shamir. How to share a secret. *Communications ACM*, 22(11):612–613, 1979.
85. K. Simonova, A. C. H. Ling, and X. S. Wang. Location-aware key predistribution scheme for wide area wireless sensor networks. In Zhu S. and Liu D., editors. SASN,. *ACM*, pages 157–168, 2006.
86. J. G. Steiner, B. C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In: *USENIX Winter*, pages 191–202, 1988.
87. D. R. Stinson. *Combinatorial Designs: Constructions and Analysis*. Springer, Berlin, 2004.
88. D. R. Stinson. *Cryptography: Theory and Practice*, 3rd edition. CRC Press, Boca Raton, FL, 2006.
89. D. R. Stinson and R. Wei. Generalized cover-free families. *Discrete Mathematics*, 279(1–3):463–477, 2004.
90. D. R. Stinson, R. Wei, and L. Zhu. Some new bounds for cover-free families. *Journal of Combination Theory Series. A*, 90(1):224–234, 2000.
91. A. P. Street and D. J. Street. *Combinatorics of Experimental Design*. Clarendon Press, Oxford, 1987.
92. R. Wei and J. Wu. Product construction of key distribution schemes for sensor networks. In Handschuh H. and Hasan M. A., editors. Selected Areas in Cryptography, volume 3357 of *Lecture Notes in Computer Science*, Springer, Berlin, pages 280–29, 2004.
93. Y. Xiao, V. K. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway. A survey of key management schemes in wireless sensor networks. *Computer Communications*, 30(11–12):2314–2341, 2007.
94. F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route filtering of injected false data in sensor networks. In: *INFOCOM*, pages 839–850, 2004.
95. M. F. Younis, K. Ghumman, and M. Eltoweissy. Location-aware combinatorialkey management scheme for clustered sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(8):865–882, 2006.
96. Z. Yu and Y. Guan. A key pre-distribution scheme using deployment knowledge for wireless sensor networks. In: *IPSN*, IEEE, pages 261–268, 2005.
97. Q. Zhang, T. Yu, and P. Ning. A framework for identifying compromised nodes in wireless sensor networks. *ACM Transactions on Information and System Security*, 11(3):12:1–12:37, 2008.
98. L. Zhou, J. Ni, and C. V. Ravishankar. Supporting secure communication and data collection in mobile sensor networks. In: *INFOCOM*, IEEE, 2006.
99. S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient security mechanisms for largescaledistributed sensor networks. In Jajodia S., Atluri V., and Jaeger T., editors. *ACM Conference on Computer and Communications Security*, ACM, pages 62–72, 2003.
100. S. Zhu, S. Xu, S, Setia, and S. Jajodia. Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach. In: *ICNP*, IEEE Computer Society, pages 326–335, 2003.

Part VIII
Tools, Applications, and Use Cases

Chapter 25
Realistic Applications for Wireless Sensor Networks

John A. Stankovic, Anthony D. Wood, and Tian He

Abstract Military surveillance, home health care or assisted living, and environmental science are three major application areas for wireless sensor networks. Revolutionary changes are possible in these application areas by using wireless sensor networks. To show the breadth and advantages of this technology, design and implementation details are presented for three systems, one in each of these three application domains. Key research challenges and the approaches taken to address them are highlighted. Challenges requiring significantly improved solutions are also identified. These systems and others like them provide significant evidence for the utility of wireless sensor networks.

25.1 Introduction

Wireless sensor networks (WSN) is an important and exciting new technology with great potential for improving many current applications in medicine, transportation, agriculture, industrial process control, and the military as well as creating new revolutionary systems in areas such as global-scale environmental monitoring, precision agriculture, home and assisted-living medical care, smart buildings and cities, and numerous future military applications [29]. In fact, it is difficult to consider any major application area that cannot benefit from WSN technology. Typically, WSN are composed of large numbers of minimal capacity sensing, computing, and communicating devices and various types of actuators. WSN operate in complex and noisy real world, real-time environments. To date, research and real-world implementations have produced many excellent low-level mechanisms and protocols to collect, transport, perform sensor fusion of this raw data, and react with control actions. This chapter discusses three implemented WSN systems which cover important application areas of wireless sensor networks: military surveillance (VigilNet [10, 34]), home medical care (AlarmNet [39]), and environmental science (Luster [27]). One goal of this chapter is to illustrate key WSN technology as

J.A. Stankovic (✉)
Department of Computer Science, University of Virginia, Charlottesville, VA, USA
e-mail: stankovic@cs.virginia.edu

required by diverse application areas. Before describing each of these systems, key overarching research challenges for WSN are briefly presented.

25.2 Challenges

Many challenging research problems must be addressed to produce a realistic WSN application. This section discusses a number of the critical challenges, specifically entitled: from raw data to knowledge, robust system operation, openness and heterogeneity, security, privacy, real time, energy management, and control (actuation). All of these challenges must deal with noisy, uncertain, and evolving environments. Each of the following three applications addresses a subset of these challenges appropriate to their purpose.

25.2.1 From Raw Data to Knowledge

Many WSNs can produce vast amounts of raw data. It is necessary to develop techniques that convert this raw data into usable knowledge in an energy efficient manner. For example, in the medical area, raw streams of sensor values must be converted into semantically meaningful activities performed by or about a person such as eating, poor respiration, or exhibiting signs of depression. Main challenges for data interpretation and the formation of knowledge include addressing noisy, physical world data, and developing new inference techniques. In addition, the overall system solution must deal with the fact that no inference method is 100% correct. Consequently, uncertainty in interpreted data can easily cause users not to trust the system. For example, in making decisions it is necessary to minimize the number of false negatives and false positives and guarantee safety, otherwise the system will be dismissed as unreliable. Location (the sensor node or base station) of the data processing is another critical issue: processing at the sensor node consumes energy and is limited by the device capacity, but it saves transmission energy and network contention. The correct trade-offs on processing location seem system dependent.

25.2.2 Robust System Operation

Many applications in wireless sensor networks typically initialize themselves by self-organizing after deployment [4]. At the conclusion of the self-organizing stage it is common for the nodes of the WSN to know their locations, have synchronized clocks, know their neighbors, and have a coherent set of parameter settings such as consistent sleep/wake-up schedules, appropriate power levels for communication, and pairwise security keys [5]. However, over time these conditions can deteriorate. The most common (and simple) example of this deterioration problem is with clock synchronization. Over time, clock drift causes nodes to have different enough times to result in application failures. While it is widely recognized that clock synchronization must re-occur, this principle is much more general. For example, even in static WSN some nodes may be physically moved unexpectedly. More and more nodes may become out of place over time. To make system-wide node locations coherent again, node re-localization needs to occur (albeit at a much slower rate than for clock sync).

These types of required coherence services must combine with many other approaches to produce robust system operation. This includes formal methods to develop reliable code, in situ debugging techniques [25], online fault tolerance [22], in-field maintenance [2], and general health monitoring services [26]. These problems are exacerbated due to the unattended operation of the system, the need for a long lifetime, the openness of the systems, and the realities of the physical world. The goal is for this collection of solutions to create a robust system [9] in spite of noisy, faulty, and non-deterministic underlying physical world realities.

25.2.3 Openness and Heterogeneity

Traditionally, the majority of sensor-based systems have been closed systems. For example, cars, airplanes, and ships have had networked sensor systems that operate largely within that vehicle. However, these systems and other WSN systems are expanding rapidly. Cars are automatically transmitting maintenance information and airplanes are sending real-time jet engine information to manufacturers. WSN will enable an even greater cooperation and two-way control on a wide scale: cars (and aircraft) talking to each other and controlling each other to avoid collisions, humans exchanging data automatically when they meet and this possibly affecting their next actions, and physiological data uploaded to doctors in real time with real-time feedback from the doctor. WSN require openness to achieve these benefits. However, supporting openness creates many new research problems including dealing with heterogeneity. All of our current composition techniques, analysis techniques, and tools need to be re-thought and developed to account for this openness and heterogeneity. New unified communication interfaces will be required to enable efficient information exchange across diverse systems and nodes. Of course, openness also causes difficulty with security and privacy, the topics of the next two sections. Consequently, openness must provide a correct balance between access to functionality and security and privacy.

25.2.4 Security

A fundamental problem that must be solved in WSN is dealing with security attacks [3, 23, 41]. Security attacks are problematic for WSN because of the minimal capacity devices being used in parts of the systems, the physical accessibility to sensor and actuator devices, and the openness of the systems including the fact that most devices will communicate wirelessly. The security problem is further exacerbated because transient and permanent random failures are commonplace in WSN and failures are vulnerabilities that can be exploited by attackers. However, the considerable redundancy in WSN creates great potential for designing them to continue to provide their specified services even in the face of failures. To meet realistic system requirements that derive from long-lived and unattended operation, WSN must be able to continue to operate satisfactorily in the presence of and to recover effectively from security attacks. The system must also be able to adapt to new attacks unanticipated when the system was first deployed.

25.2.5 Privacy

The ubiquity and interactions of WSN not only provide many conveniences and useful services for individuals, but also create many opportunities to violate privacy [15]. To solve the privacy problem created by single and interacting WSN of the future, the privacy policies for each (system) domain must be specified. Once specified, the WSN system must enforce privacy. Consequently, the system must be able to express users' requests for data access and the system's policies such that the requests can be evaluated against the policies in order to decide if they should be granted or denied. One of the more difficult privacy problems is that systems may interact with other systems, each having their own privacy policies. Consequently, inconsistencies may arise across systems. Once again, online consistency checking and notification and resolution schemes are required.

25.2.6 Real Time

Classical real-time analyses assume a rigorously defined and highly deterministic underlying system model, a set of tasks with known properties, a system that operates in a well-controlled environment, and they abstract away from properties of the physical world. For WSN systems, none of these assumptions are true and stream models rather than tasks models are prevalent. Further, WSN often support many real-time sensor streams in noisy, uncertain, and open environments. In particular, a very difficult issue is that wireless communication packet delivery is subject to burst losses. New concepts of guarantees must be developed that will likely span a spectrum from deterministic to probabilistic depending on the application, the environment, and noise and interference models [11].

25.2.7 Energy Management

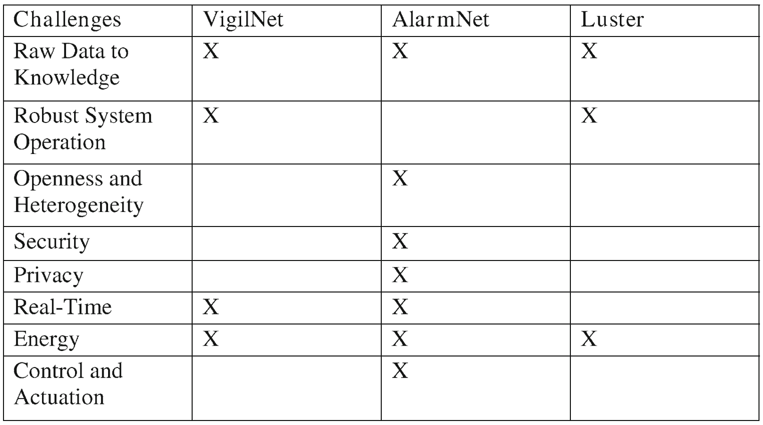
Wireless sensor networks must often operate for long periods of time. This gives rise to a significant energy management challenge. Most sensor nodes are built with the capability to control wake-sleep of each of the node's parts (cpu, memory, radio, sensors, and actuators). Many algorithms have been developed to extend the lifetime of a WSN by judiciously managing the wake–sleep for nodes and their components. In addition, many protocols found in WSN operate in a manner to minimize energy consumption. For example, MAC, routing, time synchronization, and localization protocols have been developed to function in highly efficient manners so as to extend WSN lifetimes. Recently, energy scavenging is becoming a viable addition for sensor nodes. In spite of all these solutions, energy mangement remains an important research challenge.

25.2.8 Control and Actuation

Many WSN utilize feedback control theory when actuation is involved. The classical methodology includes creating a model of the system and then deriving a controller using well-known techniques to meet stability, overshoot, settling time, and accuracy requirements. A sensitivity analysis is also performed. However, openness and scale create many difficulties for this methodology. Openness means that the model of the system is constantly changing. Human interaction is an integral aspect of openness and this makes modeling extremely difficult. In addition, scaling and interactions across systems also dynamically change the models and create a need for decentralized control. While some work has been performed in topics such as stochastic control, robust control, distributed control, and adaptive control, these areas are not developed well enough to support the degree of openness and dynamics expected in WSN. A new and richer set of techniques and theory is required. It is especially important to understand how large numbers of control loops might interact with each other. To date there have already been examples of WSN where control loops have competed with each other, one indicating an increase in a control variable while the other loop indicating a decrease in the same variable at the same time. Such dependencies must be addressed in real time and in an adaptive manner to support the expected openness of WSN.

25.2.9 Challenges and Applications

Based on application requirements, one or more of the above challenges must be addressed. Figure 25.1 identifies which of these challenges are addressed by the following three application case studies presented in the rest of the chapter.



Challenges	VigilNet	AlarmNet	Luster
Raw Data to Knowledge	X	X	X
Robust System Operation	X		X
Openness and Heterogeneity		X	
Security		X	
Privacy		X	
Real-Time	X	X	
Energy	X	X	X
Control and Actuation		X	

Fig. 25.1 Challenges and applications

25.3 Surveillance Application—VigilNet

VigilNet [10] is a military wireless sensor network that acquires and verifies information about enemy capabilities and positions of hostile targets. It has been successfully designed, built, demonstrated, and delivered to the Defense Intelligence Agency for realistic deployment. To accomplish different mission objectives, the VigilNet system consists of 40,000 lines of code, supporting multiple existing mote platforms including MICA2DOT, MICA2, and XSM. This section provides detailed technical description of the VigilNet system.

25.3.1 Application Requirements

The objective of a typical ground surveillance system is to alert the military command to targets of interest, such as moving vehicles and personnel in hostile regions. Such missions often involve a high element of risk for human personnel and require a high degree of stealthiness. Hence, the ability to deploy unmanned surveillance missions, by using wireless sensor networks, is of great practical importance for the military. Successful detection, classification, and tracking require a surveillance system to obtain the current position of a vehicle and its signature with acceptable precision and confidence. When the information is obtained, it has to be reported to a remote base station within an acceptable latency. Several application requirements must be satisfied to make this system useful in realistic environments:

- *Longevity*: Military surveillance missions typically last from a few days to several months. Due to the confidential nature of the mission and the inaccessibility of the hostile territory, it may not be possible to manually replenish the energy of the power-constrained sensor devices during the course of the mission. In addition, the static nature of the nodes in the field prevents the scavenging of the power from ambient motion or vibration. The small form factor and possible lack of the line of sight (e.g., deployment in the forest) make it difficult to harvest solar power. Hence, the application requires energy-aware schemes that can extend the lifetime of the sensor devices, so that they remain available for the duration of the mission.
- *Configuration flexibility*: It is envisioned that VigilNet will be deployed under different densities, topologies, sensing, and communication capabilities. Therefore, it is essential to design an architecture that is flexible enough to accommodate various system scenarios. For example, the system should have an adjustable sensitivity to accommodate different kinds of environment noise and security requirements. In critical missions, a high degree of sensitivity is desired to capture all potential targets even at the expense of possible false alarms. In other cases, it is desired to decrease the sensitivity of the system, maintaining a low probability of false alarms in order to avoid inappropriate actions and unnecessary power dissipation.

- *Stealthiness*: It is crucial for military surveillance systems to have a very low possibility of being detected and intercepted. Miniaturization makes sensor devices hard to detect physically; however, RF signals can be easily intercepted if sensor devices actively communicate during the surveillance stage. During the surveillance phase, a zero communication exposure is desired in the absence of significant events.
- *Real time*: As a real-time online system for target tracking, VigilNet is required to cope with fast changing events in a responsive manner. For example, a sensor node has to detect and classify a fast-moving target within a few seconds before the target moves out of the sensing range. The real-time guarantee for sensor networks is more challenging due to the following reasons. First, sensor networks directly interact with the real world, in which the physical events may exhibit unpredictable spatiotemporal properties. These properties are hard to characterize with traditional methods. Second, although the real-time performance is a key concern, it should be performance compatible with many other critical issues such as energy efficiency and system robustness. For example, the delays introduced by power management directly affect the maximum target speed VigilNet can track. It is an essential design trade-off to balance between network longevity and responsiveness.

25.3.2 *VigilNet Architecture*

The VigilNet system is designed with a layered architecture as shown in Fig. 25.2. This architecture provides an end-to-end solution for supporting military surveillance applications with multiple essential subsystems.

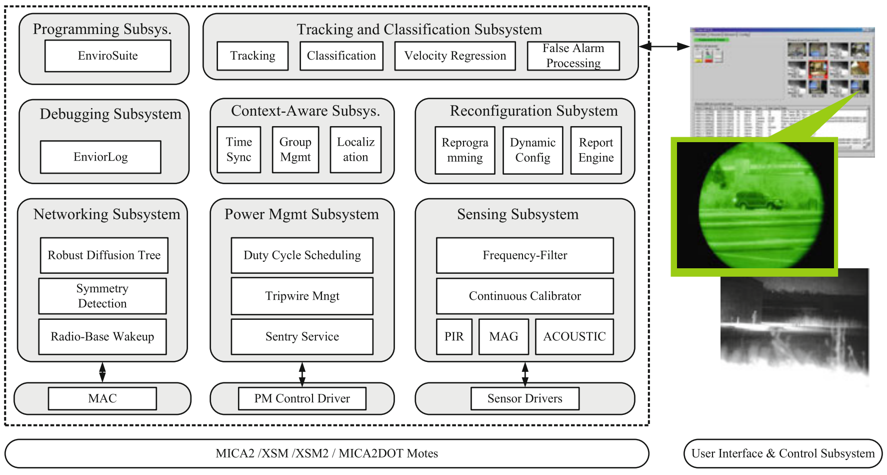


Fig. 25.2 The VigilNet system architecture

25.3.2.1 Sensing Subsystem

Sensing is the basis for any surveillance system. The VigilNet sensing subsystem implements detection and classification of targets using continuous online sensor calibration (to a changing environment) and frequency filters to determine critical target features. These filters extract the target signatures from a specific spectrum band, eliminating the burden of applying a computation-intensive fast Fourier transform. The sensing subsystem contains three detection algorithms for the magnetic sensor, acoustic sensor, and passive infrared sensor (PIR), respectively.

- The magnetic sensor detection algorithm computes two moving averages over the most recent magnetic readings. The slower moving average, with more weight on previous readings, establishes a baseline to follow the thermal drift noise caused by the changing temperature during the day. The faster moving average, with more weight on the current reading, detects the swift change in magnetic filed caused by ferrous targets. To make a detection decision, the difference between the two moving average values is compared to a dynamic threshold, which is established during the calibration phase.
- The acoustic sensor detection algorithm uses a lightweight power-based approach. It first computes a moving average of multiple recent acoustic readings, then establishes an auto-adapting acoustic threshold by calculating a moving standard deviation of readings over a certain time window. If an acoustic reading is larger than the sum of the moving average and its corresponding moving standard deviation, it is considered to be a crossover. If the number of crossovers exceeds a certain threshold during a unit of time, this algorithm signals a detection to the upper layer components.
- The passive infrared sensor is designed to sense changes in thermal radiation that are indicative of motion. When there is no movement, the thermal reading is stable and does not trigger detections. If an object is moving in front of a PIR sensor, this object causes a thermal disturbance, triggering the PIR. Most moving objects, such as shaking leaves, rain drops, and vehicles, can trigger the PIR sensor. However, different thermal signatures generate trigger events with different frequencies. Low-frequency detections (< 2 Hz) are normally triggered by wind-induced motion and other slow moving objects. On the other hand, fast-moving targets such as vehicles generate signals with a much higher frequency. Therefore, it is sufficient to design a high-pass ARMA filter to filter out the frequency components lower than 2 Hz. Since in realistic environments temperature and humidity vary significantly over the course of a day, similar to other detection algorithms, PIR's detection threshold is dynamically adjusted to accommodate such environmental changes.

25.3.2.2 Context-Awareness Subsystem

Sensed data are meaningful only when it is interpreted along with the context in which it is obtained. For example, a temperature reading is useless if it is not associated with a location and time that the value is measured. The context-awareness subsystem comprises lower level context detection components such as time synchronization and localization. These components form the basis for implementing other subsystems, such as the tracking subsystem. Localization ensures that each node is aware of its location, in order to determine the location of detected targets. Time synchronization is responsible for synchronizing the local clocks of nodes with the clock of the base station, so that every node in the network has a consistent global view of time. Combining time synchronization and localization, it is feasible to estimate the velocity of targets.

The VigilNet system uses a variation of the time synchronization protocol developed by Maroti et al. [21]. This variation eliminates the periodic time adjustments for the sake of stealthiness. To localize sensor nodes, VigilNet uses a walking GPS solution [30], in which the deployer (either person or vehicle) carries a GPS device that periodically broadcasts its location. The sensor nodes being deployed infer their position from the location broadcast by the GPS device.

25.3.2.3 Tracking and Classification Subsystem

When a target is detected by a set of nearby nodes, the tracking component creates a group. All nodes that detect the same event join a tracking group to ensure the uniqueness (one-to-one mapping of external events to logical groups) and consistent identification (immutability of the mapping function) of targets, as long as targets are far enough apart from each other or have different signatures. When targets are very near to each other and possess an identical signature, a disambiguation mechanism based on their path histories is used.

Classification is achieved through a hierarchical structure consisting of four tiers: sensor level, node level, group level, and base level. The classification result is represented by a data structure called the confidence vector. The confidence vector comprises the confidence levels for specific classes of targets and is used as a common data structure to transport information between different levels of the classification hierarchy.

- The sensor level deals with individual sensors and comprises the sensing algorithms for the corresponding sensors. With communication being an energy costly operation, the sensing algorithms need to perform local detection and classification as much as possible. After processing the sensor data, each sensing algorithm delivers the confidence vector to the higher level module — the node-level detection and classification module.
- The node-level classification deals with output from multiple sensors on the node. The fusion of the data from various sensors exposes more useful information than can be obtained from any individual sensor. Hence, the node-level sensing algorithm must correlate the sensor data from individual sensors and form node-level classification results. Such a correlation can enhance the detection and classification accuracy on individual nodes; and different sensors may strengthen the confidence of each other's classification results and invalidate false positives.

- The group-level classification is performed by groups of nodes. Such groups are managed by a middleware called EnviroSuite [17], which provides a set of distributed group management protocols to dynamically organize nodes in the vicinity of targets into groups and elect leaders among them. These leaders are designated to collect the node-level classification results from individual members and, based on them, perform the group-level classification. Thus, the input to the group-level classification is the node-level confidence vectors rather than a bulk of sample points. This greatly reduces the volume of information transmitted between group leaders and members.
- The highest level in the hierarchical classification architecture is the base-level classification. The group-level classification results are transported via multiple hops to the base station, serving as the input to the base-level classification algorithm. The base-level classification algorithm finalizes the sensing and classification result and computes various event attributes (e.g., target velocity).

25.3.2.4 Networking Subsystem

After VigilNet collects detection information about incoming targets through the tracking and classification subsystem, it needs to deliver detection reports back to the control center through a multi-hop network. The networking subsystem consists of three major components: a link symmetry detection service, a robust diffusion service, and a radio-based wakeup service. Low-power radio components, such as Chipcon CC1000 used by MICA2 [6], exhibit very irregular communication patterns. To address this problem, a link symmetry detection (LSD) module is used to reduce the impact of radio irregularity on upper layer protocols. The main idea of the LSD module is to build a symmetric overlay on top of the anisotropic radio layer, so that those protocols whose correctness depends on the link symmetry can be used without modification. More details on this solution can be found in [42]. The robust diffusion service utilizes a well-known path-reversal technique. Basically, a base node disseminates tree construction requests to the rest of the network with a running hop count initialized to zero. Requests are flooded outward with hop count incremented at every intermediate hop. After receiving tree construction requests, nodes establish multiple reverse paths toward the sending node. As a result, a multi-parent diffusion tree is constructed with the base node residing at the root. The radio-based wake-up service is designed to ensure end-to-end data delivery even if intermediate nodes are in the dormant state (due to power management). To support the illusion of on-demand wake-up, a dormant node wakes up and checks the radio activity periodically (e.g., for 5 ms every several hundred milliseconds). If no radio activity is detected, this node goes back to sleep. Otherwise, it remains active to receive and relay messages. If an active node wants to wake-up all neighboring nodes, it only needs to send out a message with a long enough preamble to last longer than the checking period of the dormant nodes.

25.3.2.5 Graphic User Interface and Control Subsystem

The networking subsystem delivers the reports to one or more command and control centers, where the graphic user interface and control subsystem is located. This subsystem provides three major functionalities. First, it accepts the reports from the sensor field and displays such information graphically to the mission operators. Second, it allows the mission operators to disseminate the system configurations through the reconfiguration subsystem. Third, based on the initial detections from the sensor field, it makes final decisions on whether to wake-up more advanced sensors. These advanced sensors not only classify the type of targets, but also differentiate the model of the targets. Since they are extremely power consuming, they are normally turned off and only used when awakened by initial detections coming from the sensor field.

25.3.2.6 The Power Management Subsystem

One of the key design objectives of the VigilNet system is to increase the system lifetime to $3 \sim 6$ months in a realistic deployment. Due to the small form factor and low-cost requirements, sensor devices such as XSM motes [7] are equipped with limited power sources (e.g., two AA batteries). The normal lifetime for such a sensor node is about 4 days if it remains active all the time. To bridge such a gap, VigilNet is equipped with a power management subsystem. Among all the middleware services, the tripwire service, sentry selection, duty cycle scheduling, and wake-up service form the basis for the power management subsystem. These services are organized into a multi-dimensional architecture. At the top level, the tripwire service is used to divide the sensor field into multiple sections, called tripwire sections. A trip wire section can be scheduled into either an active or a dormant state at a given point of time. When a trip wire section is dormant, all nodes within this section are in a deep-sleep state to conserve energy. When a trip wire section is active, a second-level sentry service is applied within this section. The basic idea of the sentry service is to select only a subset of nodes, defined as *sentries*, to be in charge of surveillance. Other nodes, defined as *non-sentries*, can be put into a deep-sleep state to conserve energy. Rotation is periodically done among all nodes, selecting the nodes with more remaining energy as sentries. At the third level, since a target can normally be sensed for a non-negligible period of time, it is not necessary to turn sentry nodes on all the time. Instead, a sentry node can be scheduled in and out of sleep state to conserve energy. The sleep/awake schedule of a sentry node can be either independent of other nodes or coordinated with that of others in order to further reduce the detection delay and increase the detection probability. More information on VigilNet power management can be found in [12].

25.3.2.7 The Reconfiguration Subsystem

The VigilNet system is designed to accommodate different node densities, network topologies, sensing and communication capabilities, and different mission objectives. Therefore, it is important to design an architecture that is flexible enough to accommodate various system scenarios. The reconfiguration subsystem addresses this issue through two major components: a multi-hop reconfiguration module and a multi-hop reprogramming module. The reconfiguration module allows fast parameter tuning through a data dissemination service, which supports limited flooding. Data fragmentation and defragmentation are supported in the reconfiguration subsystem to allow various sizes of the system parameters. The reprogramming module provides a high level of flexibility by reprogramming the nodes. More information on reprogramming can be found in [13].

25.3.2.8 The Debugging Subsystem

Debugging and tuning event-driven sensor network applications such as VigilNet are difficult for the following reasons: (i) big discrepancies exist between simulations and empirical results due to various practical issues (e.g., radio and sensing irregularity) not captured in simulators, which makes them less accurate and (ii) in-field tests of the system require walking or driving through the field to generate events of interest actively, which makes in-field tests extremely costly. To address this issue, a debugging subsystem called EnviroLog [18] is added into VigilNet. EnviroLog logs environmental events into non-volatile storage on the motes (e.g., the 512 KB external flash memory) with time stamps. These events can then be replayed in their original time sequence on demand. EnviroLog reduces experimental overhead by eliminating the need to physically re-generate events of interest hundreds of times for debugging or parameter tuning purposes. It also facilitates comparisons between different evaluated protocols.

25.3.3 The Programming Interface

The programming interface in VigilNet is an extension of the prior work on EnviroSuite [17]. It adopts an object-based programming model that combines logical objects and physical elements in the external environment into the same object space. EnviroSuite differs from traditional object-oriented languages in that its objects may be representatives of physical environmental elements. EnviroSuite makes such objects the basic computation, communication, and actuation unit, as opposed to individual nodes. Thus, it hides implementation details such as individual node activities and interactions among nodes. Using language primitives provided by EnviroSuite, developers of tracking or monitoring applications can simply specify object creation conditions (sensory signatures of targets), object attributes (monitored aggregate properties of targets), and object methods (desired computation, communication, or actuation in the vicinity of targets). Such specifications can be translated by an EnviroSuite compiler into real applications that are directly executable on motes. When defined object conditions are met, dynamic object instances are automatically created by the runtime system of EnviroSuite to collect object attributes and execute object methods. Such instances float across the network following the targets they represent and are destroyed when the targets disappear or move out of the network.

25.3.4 System Work Flow

To avoid interference among different operations, VigilNet employs a multiple-phase work flow. The transition between phases is time driven, as shown in Fig. 25.3. Phases I through VII comprise the initialization process which normally takes about several minutes. In phase I, the reconfiguration subsystem initializes the whole network with a set of parameters. In phase II, the context-awareness subsystem synchronizes all nodes in the field with the master clock at the base, followed by the localization process in phase III. In phases IV and V, the networking subsystem establishes a robust diffusion tree for end-to-end data delivery. Phase VI invokes the power management subsystem to activate trip wire sections and select a subset of the nodes as sentries. The system layout, sentry distribution, and network topology are reported to the graphic user interface and control subsystem in phase VII. After that, the nodes enter into the main phase VIII—the surveillance phase. In this phase, nodes enable the power management subsystem in the absence of significant events and activate the tracking subsystem once a target enters into the area of interest.

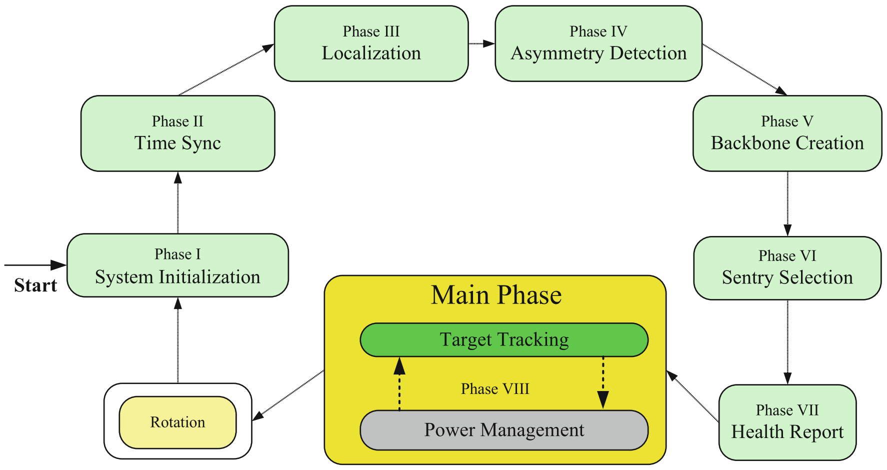


Fig. 25.3 Phase transition and rotation

25.3.5 VigilNet Summary

Surveillance using wireless sensor networks is a very practical application. It has many advantages such as fast ad hoc deployment, fine-grained robust sensing and tracking, low-power consumption, and low cost. From the experience in building

VigilNet, it is clear that realistic issues must not be ignored in developing usable solutions. This includes sensor failures, environmental changes, asymmetries in communication, and false alarms. Because of these issues, debugging and performance tuning in distributed sensor networks are extremely time consuming, especially during field tests. Therefore, it is critical to have appropriate built-in system support for these functions, such as the reconfiguration subsystem and the debugging subsystem. Since the sensor nodes fail at a much higher rate in hostile outdoor environments, self-healing should be supported by every protocol integrated into the system. Despite these realistic challenges faced, VigilNet presents a proof that viable surveillance systems can be implemented and successfully deployed on low-power sensor devices.

25.4 Healthcare Applications—AlarmNet

In this section the AlarmNet system is described. AlarmNet is an assisted-living and residential monitoring network for pervasive, adaptive health care in assisted-living communities with residents or patients with diverse needs. The system contains over 15 types of sensor nodes (pulse ox, ECG, temperature, etc.), runs on TelosBs, MicaZs, PDAs, and PCs. Various parts of AlarmNet have been described before [8, 28, 36, 39, 40], but here the focus is on the following aspects:

- An extensible, heterogeneous network middleware that addresses the challenges of an ad hoc wide-scale deployment and integrates embedded devices, back-end systems, online analysis, and user interfaces;
- SenQ, a query protocol for efficiently streaming online sensor data to the system and to users, integrated with privacy, power management, and activity analysis, and
- Novel context-aware protocols using two-way network information flow: environmental, system, and resident data flow into the back end, and circadian activity rhythm analysis feeds back into the system to enable smart power management and dynamic alert-driven privacy tailored to an individual's activity patterns.

25.4.1 Application Requirements

An aging baby-boom generation is stressing the US healthcare system, causing hospitals and other medical caregivers to look for ways to reduce costs while maintaining quality of care. It is economically and socially beneficial to reduce the burden of disease treatment by enhancing prevention and early detection. This requires a long-term shift from a centralized, expert-driven, crisis-care model to one that permeates personal living spaces and involves informal caregivers, such as family, friends, and community.

Systems for enhancing medical diagnosis and information technology often focus on the clinical environment and depend on the extensive infrastructure present in traditional healthcare settings. The expense of high-fidelity sensors limits the number that are available for outpatient deployment and some requires specialized training to operate. Manual record keeping has been identified as a key source of medical errors [24], and at its best, traditional data collection is intermittent, leaving gaps in the medical record.

Wireless sensor networks (WSNs) provide capabilities that are valuable for continuous, remote monitoring, as research into military and environmental systems attest. For healthcare applications, they can be deployed inexpensively in existing structures without IT infrastructure. Data are collected automatically, enabling daily care and longitudinal medical monitoring and diagnosis. The wireless devices can integrate with a wide variety of environmental and medical sensors.

While addressing some of the needs of distributed health care, WSNs also present their own challenges, both practical and theoretical, to being robust platforms for pervasive deployment. Privacy and security of collected medical data may be jeopardized by careless use of a wireless medium. Without smart power management, battery-powered sensors have short lifetimes of a few days or require continual maintenance.

25.4.2 AlarmNet Architecture

A key requirement for healthcare systems is the ability to operate continuously over long time periods and still integrate new technologies as they become available. AlarmNet satisfies these objectives by unifying and accommodating heterogeneous devices in a common architecture (see Fig. 25.4) that spans wearable body networks, emplaced wireless sensors, user interfaces, and back-end processing elements.

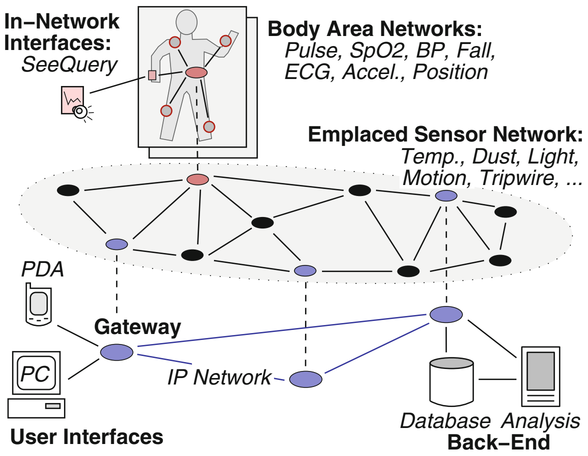


Fig. 25.4 Multi-tier AlarmNet architecture with emplaced sensors, mobile body area networks, a backbone of gateways, user interfaces, and back-end storage and processing

Mobile body networks are wireless sensor devices worn by a resident which provide activity classification or physiological sensing, such as an ECG, pulse oximeter, or accelerometers. The body network is tailored to the patient's own medical needs and can provide notifications (e.g., alerts to take medicine) using an in-network wearable interface (e.g., the SeeMote [28] that has a color LCD). □

It also integrates SATIRE [8], a body network that classifies activities of daily living (ADLs) [16] by analyzing accelerometer data generated by a wearer's movements.

Body networks contain a designated gateway device that mediates interaction with the surrounding WSN. This modularizes the system's interaction with the body network to ease its integration. Data are streamed directly or multi-hop through the emplaced network to the AlarmGate gateways for storage, analysis, or distribution to user interfaces.

Emplaced sensors are deployed in living spaces to sense environmental quality, such as temperature, dust, and light, or resident activities. Motion and trip wire sensors, in particular, provide a spatial context for activities and enable location tracking.

Due to their low cost, small form factor, and limited power budget, the devices answer queries for local data and perform limited processing and caching. Though some deployment environments may enable the use of wired electrical power, it is not required so as to support ad hoc retro-fitting of existing structures. Figure 25.5a shows the lightweight stack resident on sensor devices.

AlarmNet supports dynamically adding new devices to the network, which register their capabilities and are initialized. This flexibility allows the system to change over time as sensors are developed or new pathologies require monitoring.

AlarmGate applications run on an embedded platform, such as the Crossbow stargate, and serve as a communication backbone and application-level gateway between the wireless sensor and IP networks. Owing to their greater resources, these devices perform major aspects system operation related to dynamic privacy, power management, query management, and security. The AlarmGate software stack is shown in Fig. 25.5b.

Back-end programs perform online analysis of sensor data, feeding back behavior profiles to aid context-aware power management and privacy. A database provides

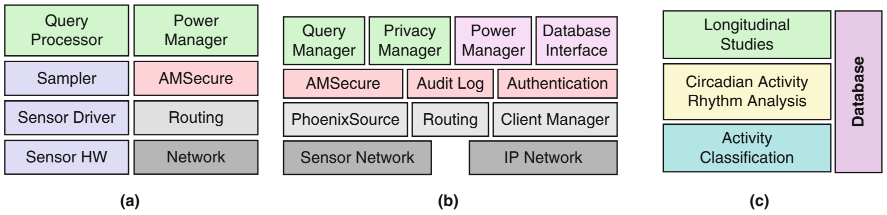


Fig. 25.5 AlarmNet software architecture. (**a**) Embedded software stack on sensor devices. (**b**) AlarmGate software stack on network gateways. (**c**) Back-end analysis and persistent storage

long-term storage of system configuration, user information, privacy policies, and audit records.

One such program, for circadian activity rhythm (CAR) analysis, processes sensor data stored in the database and learns behavior patterns of residents. These are used to detect deviations from personal norms that may signal a short- or long-term decline in resident health.

The back end is extensible to new analyses using a modular framework, wherein programs consume input sensor streams, filter and process them, and produce output streams in the database for other modules to use. These are composed hierarchically from low-level sensor streams to high-level inference of symptoms and diseases.

User interfaces allow doctors, nurses, residents, family, and others to query sensor data, subject to enforced privacy policies. We developed a patient-tracking GUI for a nurse's station and a query issuer for a PDA that graphs sensor data in real time. These programs are not trusted components—they must connect through AlarmGate and do not have direct access to the database. This makes it easier to develop and deploy new interfaces customized to the application's needs.

In summary, AlarmNet's architecture supports health-monitoring applications due to its flexibility and extensibility in (1) supporting dynamic addition of heterogeneous devices, sensors, and body networks, (2) feeding learned resident and system context back into the network, and (3) providing an open client model for future extension.

25.4.3 Query Management

A primary reason for developing AlarmNet was to use environmental, physiological, and activity data of assisted-living residents to improve their health outcomes. The automated analysis programs need to automatically collect data in the background, but the system must also support ad hoc queries by healthcare providers and the addition of new analysis programs over time. Existing data management solutions were mostly optimized for tree-based aggregation [19] or else used general-purpose virtual machines for arbitrary computation.

SenQ is a query system that satisfies the requirements of the healthcare domain: reconfigurable in-network sensing and processing, dynamic query origination by embedded devices, and high-level abstractions for expressing queries. A detailed treatment of SenQ's lower layers and their performance was given in [40], with focus on sensing and in-network query processing. Here the system-level query management functions and their integration with other components of AlarmNet are presented.

The back-end system, user interfaces, and embedded devices all issue queries using a common network protocol, in which queries are uniquely identified by *<source ID, query ID>* tuples. Users may request a snapshot of the current value or a periodic stream of a sensing modality. To reduce repetitive query parsing overhead on resource-constrained motes, both types of queries may be cached and efficiently restarted (or reissued) later.

Since radio communication in the WSN is expensive, it is desirable to process data at its source, if possible, to reduce the amount that is reported. However, sensor devices have limited memories and processors, and so only relatively lightweight processing is practical. SenQ dynamically constructs a scalar processing chain on the mote to perform spatial and temporal aggregation and filtering to reduce the energy consumed by communication.

Query manager is a major actor in the query subsystem and resides on the gateway, in the AlarmGate software. Devices are commonly added to and removed from the system, particularly in the healthcare domain where monitoring needs evolve over time. To enable the query manager to maintain device state, nodes *register* with the nearest gateway upon power-up, providing their device type, sensors, and hardware ID. They are assigned dynamic network IDs and localized via application-specific means.

The query manager issues background queries to devices as they are added to the network to satisfy the system's core management and tracking functionalities. Examples of background queries in AlarmNet are

- All devices sample and report their battery supply voltage every 4 h, but only if it is below 2.8 V (indicating imminent failure);
- Motion, trip-wire, and contact-switch sensors report activations on demand, but no more often than every 100 ms to debounce or dampen spurious bursts;
- Pulse oximetry devices, which are intermittently switched on, collect heart rate and SpO_2 samples every 250 ms, but report them every 750 ms, each an average of three samples until the device is switched off; and
- ECG sensors immediately begin reporting a stream of raw samples every 20 ms, using full buffering to reduce network load and energy usage.

The query manager is the main point of access for *user interfaces*, translating between higher level query abstractions and the SenQ protocol exchanged with particular sensor devices. Connected users receive a list of active devices that is updated in real time as registrations are received. However, most users of the system will not have detailed knowledge of its current topology. Usability is improved if they can request sensor data semantically for people and locations. This presents a few challenges for query management on the gateway.

A request for sensor information about person P must be mapped to a device (or group of devices) D for execution. Some have static associations, such as a wearable device owned or assigned to a user. Likewise for locations L in which fixed sensor nodes are placed. But since networks for assisted living are more human oriented and heterogeneous than most other types of WSNs, many sensor types require dynamic binding based on a person's context (location, activity, etc). Externalizing such bindings, as in other common approaches for WSNs, results in duplicated effort to track user and device state by both the query system and the application that uses it.

Dynamic semantic binding simplifies data access for users, but the challenge is how to provide it in a modular way that does not limit SenQ to a particular deployment environment. Our approach is to share the core of a context model, shown in

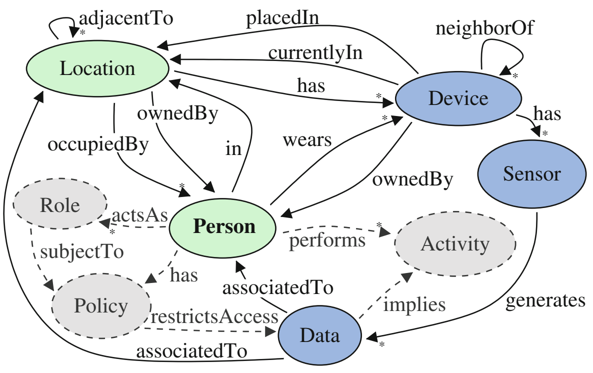


Fig. 25.6 SenQ's context model. AlarmNet extensions are in grey

Fig. 25.6, with applications co-resident on the query management gateway. Relations among the core components (device, sensor, location, person, and data) are maintained by SenQ and consulted for mapping queries to devices.

Instantiations of the system extend the model as appropriate; for example, AlarmNet adds privacy, power management, and activity analysis. These and any future extensions can access and mutate the core shared context to change SenQ's bindings of queries to sensors. AlarmNet's context-aware modules are described next, beginning with activity analysis which also interacts with the others.

25.4.4 Circadian Activity Rhythms

It is known that most people exhibit behavioral trends in the home with 24-h cycles are called "circadian activity rhythms" (CAR). AlarmNet includes a CAR analysis program that measures the rhythmic behavioral activity of residents and detects changes within these patterns.

The CAR algorithm is statistical and predictive and was first presented in [35]. Here we present its integration with AlarmNet and some recent extensions. CAR is used to improve both medical care and network performance. In particular, CAR supports context-aware protocols based on learned activity patterns for smart heterogeneous power management and dynamic alarm-driven privacy.

Rhythms based on an hourly distribution of the probability of user presence in every room are called *presence-based CAR*. Those based on the density of the number of events per hour are called *activity level-based CAR*. The CAR analysis program runs on a PC on the back end of the system and reads a database of resident activity.

The CAR program provides a GUI to display various information related to the activity analysis, such as the number of abnormal time periods (under-presence or over-presence in a room for presence-based CAR), the degree of

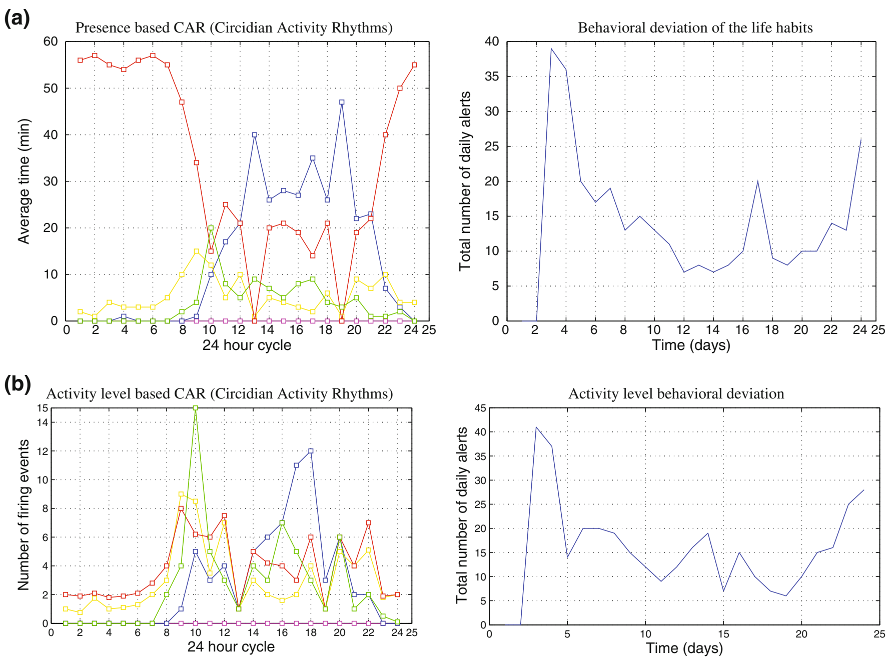


Fig. 25.7 Circadian activity rhythm (CAR) analysis GUIs. Sums of daily deviations from the user's norm are on the right side of each GUI, showing a learning period after initial deployment (**a**) Presence based CAR analysis GUI; □ average time spent in every room per hour is graphed on the left side. (**b**) Activity level based CAR analysis GUI; □ average number of motion sensor events in every room per hour is on the left side. Each colored curve is a different room in the house

activity that occurred per hour and day during day or night (hypo or hyper-activity level for activity level-based CAR), and the length and dates of stay of the resident.

Other graphs of the GUI display the main results of the CAR analysis. The graphs in Fig. 25.7 present data from a clinical case study for a healthy resident who stayed 25 days in an assisted-living facility. The first one (Fig. 25.7a) displays the average time the user spends in every room each hour, calculated over the number of days of the stay of the resident. On the right side, the graph indicates deviations in room presence. The experiments demonstrated that the CAR program needed a period of approximately 2 weeks to learn normal behavior patterns. The graphs in Fig. 25.7b represent the same experiment (same subject, same clinical protocol, and same period of study), but for his activity levels.

Comparing both graphs gives complementary information concerning normal and abnormal activity levels in the different rooms of the dwelling. These graphs can provide a wealth of information about activity patterns such as the sleep/wake cycle or some medical hints to the physician about some activities of daily living (ADLs) [16] of the resident such as eating, hygiene, and sleeping. In the future, more specific ADLs will also be inferred.

After the learning period, any statistically significant deviations from learned patterns are displayed as alerts on a GUI and are sent to the AlarmGate application. Nurses or physicians can investigate the source of the trouble by focusing on the region of the anomaly as identified by CAR □. The hypothesis is that deviations from a resident's own in-home behaviors can support medical diagnosis. For example, increased sleep time and multiple missed meals may indicate illness in the short-term or reveal overall decline in the health of the resident in the long term.

This hypothesis was investigated clinically in collaboration with the Medical Automation Research Center (MARC) at the University of Virginia School of Medicine. Clinical behavioral patterns of older adults in assisted-living facilities were extracted from real data sets, and behavioral changes were studied by consulting the medical notebooks of the caregivers in charge of the monitored residents. The capability to detect anomalies from the norm calculated over multiple days was proved [36], but inferring pathologies or onsets of chronic pathologies remains to be explored.

25.4.5 Dynamic Context-Aware Privacy

Data collected in AlarmNet reveals intimate details about a person's life activities and health status. As WSNs grow stronger in their capability to collect, process, and store data, personal information privacy becomes a rising concern. AlarmNet includes a framework to protect privacy and still support timely assistance to residents in critical health situations.

Emergency-aware applications demand a privacy protection framework capable of responding adaptively to each resident's health condition and privacy requirements in real time. Traditional role-based access control, which makes access authorization decisions based on users' static roles and policies, is not flexible enough to meet this demand. A privacy protection framework was designed which is dynamically adjustable to residents' context, allows data access authorization to be evaluated at runtime, and is able to adapt to residents' health emergencies.

A key novelty is that access rulings are dynamically altered based on context inferred by the CAR and other back-end programs, when necessary. For example, if a resident has blocked access to his ECG data for nurses, but the CAR has detected serious anomalous behavior that might indicate a heart problem, then a nurse is alerted and access to the data is allowed for a period of time.

In case of an alarming health status, privacy may be restricted or relaxed depending on context and who requests the data. This decision is recorded in the database with details of the mitigating context. It can be used to notify residents later about what transpired and why and who accessed their data during the emergency.

Other context in our system includes the resident's physiological condition (ECG, pulse, blood pressure), living environment conditions (room temperature, light levels, dust), activities, and autonomy (inferred from ADLs by the CAR). □

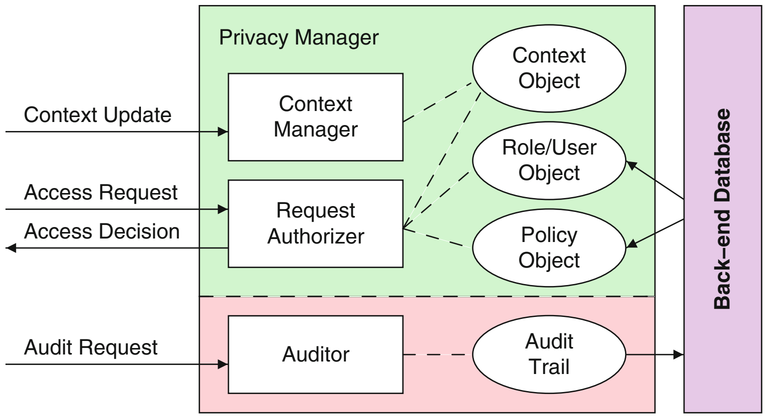


Fig. 25.8 Privacy-related components in AlarmNet

System designers may specify privacy policies at different levels of granularity, from individual sensors to residents or groups of residents.

The privacy management framework resides in the AlarmGate application and has three main functional components: the context manager, the request authorizer, and the auditor (shown in Fig. 25.8).

Context manager, as described before, maintains residents' current locations, activities, and health conditions. In the privacy module, context is indexed by the tuple *<context id, context subject, context value>*.

Request authorizer is consulted when data queries are received at the query manager. It makes access decisions by consulting the system's privacy policies and the context of the query subject. After each access request is decided, it is recorded by the auditor module.

SenQ allows queries for locations and devices, which must be mapped to a resident for authorization by the request authorizer. Locations may be assigned an owner, such as the resident living in a particular unit. Common areas without an owner use a default subject or policy. Devices also may have an owner, for example, an ECG that is worn by a particular resident. Otherwise, the device's assigned or current location is used to determine the context subject.

Auditor maintains a trace of access requests in an audit trail, including the authorization decision made for each request (granted or denied). Dynamic decisions during alarming situations trigger the storage of details of the context for later notification of residents and investigation of misuse. Such records may be crucial in heavily regulated environments like health care.

25.4.6 AlarmNet Summary

AlarmNet is a wireless sensor network system designed for long-term health monitoring in assisted-living environments with two central design goals. First, system operation adapts to the individual context and behavior patterns of the residents, which feeds back to influence power management and privacy policy enforcement. Second, the system is extensible and supports a diverse collection of sensors, user interfaces, and power and privacy policies. Systems such as AlarmNet and the proliferation of individual wireless medical devices are having a transformative effect on medical care. It is being referred to as wireless medicine.

25.5 Environmental Science Applications—Luster

One of the most beneficial and interesting applications of WSNs is the ability to create a "macroscope"—to take a look at the big picture of the monitored environment. There have been many implementations of macroscopes, for example, a WSN deployed on redwood trees [33], a wildlife monitoring site on Great Duck Island [20, 31, 32], tracking zebras in their natural habitat [14], and monitoring volcanic eruptions [37, 38].

In this section, Luster, an environmental science application for measuring the effect of sunlight on under shrub growth on barrier islands is described. Luster is built with custom light sensor boards that support up to eight light sensors each and these boards are attached to MicaZs. Luster also uses a custom built SeeMote [28] that contains an SD card with 2 GB of memory.

The main features of Luster are

- *An overlay network architecture*—A hierarchical structure for sensing, communication, and storage allows replication of the system in clusters for scalability.
- *Reliable, transparent, and distributed storage*—Fault-tolerant storage is provided by unobtrusively listening to sensor node communications, thus minimizing power requirements without the need of dedicated queries.
- *Delay-tolerant networking*—Access is provided to the measured data over an unreliable or intermittent network connection.
- *Custom hardware*—New hardware designs provide combined sensing and energy harvesting, removable storage, and lightweight in situ WSN interfaces for deployment time validation.
- *Deployment time validation*—New techniques and tools were developed to increase the probability of successful deployment and long-term operation of the system by looking for problems and fixing them early.
- *Customizable web interface*—Remote access to the WSN and visualization of sensor data are provided using reconfigurable hypertext templates.

25.5.1 Application Requirements

Shrub cover is increasing worldwide, most often by replacing herbaceous-dominated communities, especially grasslands [1]. In North America, this phenomenon has been documented in a range of environments from desert grasslands and arctic tundra to Atlantic coast barrier islands. Shrub encroachment has been linked to climate change and anthropogenic factors; however, the causes have not been thoroughly investigated. Perhaps most dramatic is the change in the light environment as community structure shifts from grassland to shrubland. Most studies measure light either at a single point in time or over periods of minutes or hours. However, there is a need to quantify light at finer temporal scales because photosynthetic responses occur within seconds and courser measurements often underestimate the effects of brief, but intense, direct light. Measuring light at a fine spatial and temporal granularity is the primary purpose of the Luster system.

Luster addresses many WSN issues including the harsh elements of nature that cause rapid device and sensor malfunction and network links to back-end monitoring and collection systems that may be intermittent due to weather or other problems.

25.5.2 Luster's Architecture

As can be inferred from the above list of capabilities, the primary challenges addressed by Luster are robust system operation, openness, and real time. To address these challenges, Luster's architecture is composed of several layers, shown in Fig. 25.9. The *sensor node layer* is responsible for gathering, aggregating, and transmitting the measurement data. The report rate and sensor selection are configurable remotely using the SenQ sensor network query system [39]. Communication among nodes in the cluster uses LiteTDMA, a novel MAC protocol developed specifically to address the robust and real-time aspects of the system.

The *storage layer* transparently blankets the sensor layer, collecting and filtering the data reported by sensor nodes without initiating any communication to them. Thus, bandwidth and power consumption are improved. The configuration specifies the data filtering and collection policies for each storage node, alleviating congestion internal to the storage hardware due to the flash memory delays. Reliability is provided through redundant coverage: each sensor node is monitored by at least

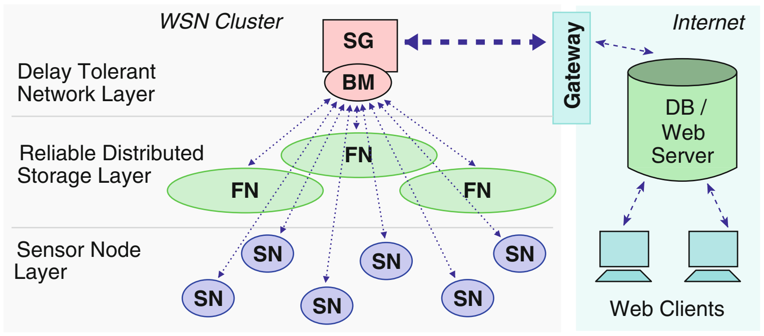


Fig. 25.9 Luster's hierarchical architecture

two storage nodes. The configuration and the number of storage nodes are subject to application requirements for the fault tolerance required and cost limitations.

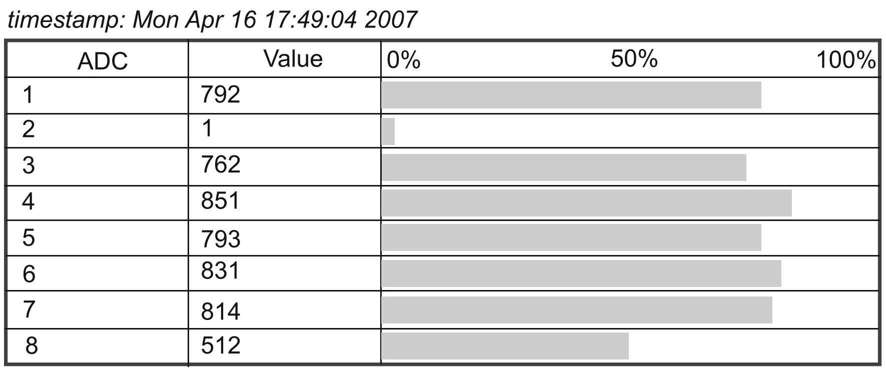
Above the storage layer is the *delay-tolerant networking* (DTN) component of the system. This consists of a base mote (denoted BM in Fig. 25.9) attached to a stargate (SG) acting as a gateway between the IEEE 802.15.4 and IEEE 802.11 networks. Absolute reliability of this communication link is not assumed. Instead, distributed storage is used to capture all the data and either serve it in response to queries after the communication link between the WSN and the Internet is reestablished or in a delayed fashion by collecting the removable storage cards during a subsequent visit to the deployment area. The latter is the most energy efficient method for the WSN, especially when many sensors are required.

The data monitored by the WSN is accessible over the Internet by connecting to the Luster *back-end server*. The server stores the incoming WSN data stream to a database and also issues data queries to the WSN as a DTN component detects losses.

Using a web browser, a user can request and view historical data as well as the last captured "almost real-time" data from the back-end server. Web pages are generated from HTML templates, in which the WSN data, including the sensor readings, are embedded in the HTML as custom tags. This allows for a user-centric customizable web interface that is specific to the chosen application.

An example of an application-neutral display is shown in Fig. 25.10, in which a node's current sensor readings have been requested. Values from the eight ADC channels are tabulated and graphed. Using custom HTML tags, a user can add semantically meaningful labels for the channels and their values. Near real-time display of sensor readings supports online diagnostics to determine, for example, that ADC channel two in Fig. 25.10 is faulty and stuck at a low value.

For *scalability*, the WSN architecture described (and shown on the left side of Fig. 25.9) is replicated into multiple clusters that cover the entire deployment area. Clusters use single-hop communication internally, and are adjoined and managed by the stargate node using several techniques.



timestamp: Mon Apr 16 17:49:04 2007

ADC	Value	0% 50% 100%
1	792	
2	1	
3	762	
4	851	
5	793	
6	831	
7	814	
8	512	

Fig. 25.10 Luster web server generated page reporting ADC levels for a pre-selected remote sensor node

First, each cluster is assigned a communication channel that avoids or minimizes interference with its neighbors. One challenge with a multiple channel approach is the likelihood of the base mote missing communications on one channel while listening to another. This challenge is met by the delay-tolerant design of Luster, which operates even when communications from the WSN clusters to the main back-end server are lost. Distributed storage and delay-tolerant networking components allow recovery of data as described above.

A second approach for inter-cluster interference minimization is to interleave the LiteTDMA MAC communication schedules so that when one cluster is communicating, the others are asleep. The stargate node acts as a super-master to coordinate the schedules. Finally, the transmission power of nodes can be adjusted to match the cluster's coverage area.

25.5.3 Luster Summary

Many environmental science applications have been implemented with WSN technology. Many of them have similar requirements to Luster such as remote access, reliable data collection and storage, and real-time viewing. There is often a need for a specialized network, e.g., in Luster the coverage area for each of the 124 sensors is on the order of meters allowing a single-hop local network based on a TDMA access control protocol. In other cases, e.g., Zebranet, there is a requirement to support mobility. In the future, as the underlying WSN technology matures, there should be more emphasis placed on creating knowledge from the data.

25.6 Summary

Wireless sensor networks are entering a second generation. The first generation has seen many new research challenges being defined and solved. This has resulted in many interesting systems being implemented. Three of these systems, VigilNet (for military surveillance), AlarmNet (home health care and assisted living), and Luster (environmental science) are described in this chapter as representative of this first generation. New challenges have emerged for second generation wireless sensor networks including: creating knowledge from raw data, robust system operation, dealing with openness and heterogeneity, security, privacy, real-time, and control and actuation. This is not a complete list of challenges, but these do constitute some of the major challenges as wireless sensor networks become widespread and move into many other application domains such as agriculture, energy, and transportation.

Acknowledgments We thank all the students that worked on one or more of these systems. The work was supported, in part, by NSF grants CNS-0614870, CNS-0614773, CNS-0626616, CNS-0626632, the DARPA NEST project and a DoD MURI.

References

1. S. T. Brantley and D. R. Young. Leaf-area index and light attenuation in rapidly expanding shrub thickets. *Ecology*, 88(2):524–530, 2007.
2. Q. Cao and J. A. Stankovic. An in-field maintenance framework for wireless sensor networks. In: *Proceedings of DCOSS*, 2008.
3. S. Capkun and J. P. Hubaux. Secure positioning of wireless devices with application to sensor networks. In: *Proceedings of IEEE INFOCOM*, 2005.
4. A. Cerpa and D. Estrin. Ascent: Adaptive self-configuring sensor networks topologies. In: *Proceedings of INFOCOM*, 2002.
5. H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In: *IEEE Symposium on Research in Security and Privacy*, pages 197–213 (May 2003).
6. CrossBow. *Mica2 data sheet*. Available at http://www.xbow.com (August 2009).
7. P. Dutta, M. Grimmer, A. Arora, S. Biby, and D. Culler. Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In: *IPSN'05*, 2005.
8. R. K. Ganti, P. Jayachandran, and T. F. Abdelzaher. SATIRE: A software architecture for Smart AtTIRE. In: *4th ACM Conference on Mobile Systems, Applications, and Services (Mobisys 2006)*, 2006.
9. L. Gu and J. A. Stankovic. t-kernel: Providing reliable os support for wireless sensor networks. In: *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys'06)*, 2006.
10. T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. Stankovic, T. Abdelzaher, J. Hui, and B. Krogh. Vigilnet: An integrated sensor network system for energy efficient surveillance. *ACM Transactions on Sensor Networks*, 2(1):1–38 (February 2006).
11. T. He, P. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q. Cao, J. Stankovic, and T. Abdelzaher. Achieving real-time target tracking using wireless sensor networks. *ACM Transactions on Embedded Computing System*, 30(5):627–633 (October 2008).
12. T. He, P. Vicaire, T. Yan, Q. Cao, G. Zhou, L. Gu, L. Luo, R. Stoleru, J. A. Stankovic, and T. Abdelzaher. Achieving long-term surveillance in VigilNet. In: *The 25th Conference on Computer Communications (INFOCOM 2006)*, pages 1–12 (April 2006).
13. J. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In: *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)* (November 2004).
14. P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In: *Proceedings of ASPLOS-X*, pages 96–107, 2002.
15. P. Kamat, Y. Zhang, W. Trappe, and C. Osturk. Enhancing source location privacy in sensor network routing. In: *Proceedings of International Conference on Distributed Computing Systems*, 2005.
16. S. Katz, A. B. Ford, R. W. Moskowitz, B. A. Jackson, and M. W. Jaffe. Studies of illness in the aged. The index of adl: A standardized measure of biological and psychosocial function. *Journal of the American Medical Association*, 185:914–919 (September 1963).
17. L. Luo, T. Abdelzaher, T. He, and J. A. Stankovic. EnviroSuite: An environmentally immersive programming framework for sensor networks. *ACM Transactions on Embedded Computing System, Special issue on Dynamically Adaptable Embedded Systems*, 5(3):21:1–35 (August 2006).
18. L. Luo, T. He, G. Zhou, L. Gu, T. Abdelzaher, and J. Stankovic. Achieving repeatability of asynchronous events in wireless sensor networks with EnviroLog. In: *The 25th Conference on Computer Communications (INFOCOM 2006)*, pages 1–14 (April 2006).
19. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: An acqusitional query processing system for sensor networks. *ACM TODS*, 30(1):122–173, 2005.

20. A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In: *Proceedings of WSNA*, pages 88–97, 2002.
21. M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In: *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)* (November 2004).
22. L. Paradis and Q. Han. A survey of fault management in wireless sensor networks. *Journal of Network and Systems Management*, 15(2):171–190, 2007.
23. A. Perrig, J. Stankovic, and D. Wagner. Security in wireless sensor networks. *Communications of the ACM*, 47(6):53–57, 2004.
24. President's Information Technology Advisory Committee (PITAC), *Revolutionizing Health Care Through Information Technology*, National Coordination office, Arlington, VA (June 2004).
25. N. Ramanathan, K. Chang, L. Girod, R. Kapur, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. In: *Proceedings of the 3rd ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.
26. S. Rost and H. Balakrishnan. Memento: A health monitoring system for wireless sensor networks. In: *Proceedings of IEEE SECON*, 2006.
27. L. Selavo, A. Wood, Q. Cao, A. Srinivasan, H. Liu, T. Sookoor, and J. Stankovic. Luster: Wireless sensor network for environmental research. 5th ACM Conference on Embedded Networked Sensor Systems, Sydney, Australia (November 2007).
28. L. Selavo, G. Zhou, and J. A. Stankovic. SeeMote: In-situ visualization and logging device for wireless sensor networks. In: *Proceedings of BASENETS*, 2006.
29. J. Stankovic. When sensor and actuator networks cover the world. *ETRI Journal*, 30(5):627–633 (October 2008).
30. R. Stoleru, T. He, and J. A. Stankovic. Walking GPS: A practical solution for localization in manually deployed wireless sensor networks. In: *1st IEEE Workshop on Embedded Networked Sensors EmNetS-I* Tampa, Fla (October 2004).
31. R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In: *Proceedings of SenSys*, pages 214–226, 2004.
32. R. Szewczyk, J. Polastre, A. M. Mainwaring, and D. E. Culler. Lessons from a sensor network expedition. In H. Karl, A. Willig, and A. Wolisz, editors. *EWSN*, volume 2920 of *Lecture Notes in Computer Science*. Springer, Berlin, pages 307–322, 2004.
33. G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In: *Proceedings of SenSys*, pages 51–63, 2005.
34. P. Vicaire, T. He, T. Yan, Q. Cao, G. Zhouand L. Gu, L. Luo, R. Stoleru, J. Stankovic, and T. Abdelzaher. Achieving long term surveillance in vigilnet. *IEEE Computer*, 5(1):1–39 (January 2009).
35. G. Virone, N. Noury, and J. Demongeot. A system for automatic measurement of circadian activity deviation in telemedicine. *IEEE Transactions on Biomedical Engineering*, 49(12):1463–1469 (December 2002).
36. G. Virone, M. Alwan, S. Dalal, S. Kell, B. Turner, J. A. Stankovic, and R. Felder. Behavioral patterns of older adults in assisted living. *Information Technology in Biomedicine*, 12(3):387–398 (May 2008).
37. M. Welsh, G. Werner-Allen, K. Lorincz, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees. Sensor networks for high-resolution monitoring of volcanic activity. In: *Proceedings of SOSP*, pages 1–13, 2005.
38. G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, 2006.
39. A. Wood, J. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, and R. Stoleru. Context-aware wireless sensor networks for assisted living and residential monitoring. *IEEE Network*, 22(4):26–33, 2008.

40. A. D. Wood, L. Selavo, and J. A. Stankovic. SenQ: An embedded query system for streaming data in heterogeneous interactive wireless sensor networks. In: *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, volume 5067 of *Lecture Notes in Computer Science*, pages 531–543. Springer, Berlin, 2008.
41. A. D. Wood, and J. A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62 (October 2002).
42. G. Zhou, T. He, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In: *The Second International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2004.

Chapter 26
High-Level Application Development for Sensor Networks: Data-Driven Approach

Animesh Pathak and Viktor K. Prasanna

Abstract Owing to the large scale of networked sensor systems, ease of programming remains a hurdle in their wide acceptance. High-level application development techniques, or *macroprogramming* provides an easy-to-use high-level representation to the application developer, who can focus on specifying the behavior of the *system*, as opposed to the *constituent nodes* of the wireless sensor network (WSN). This chapter provides an overview of the current approaches to high-level application design for WSNs, going into the details related to data-driven macroprogramming. Details of one such language are provided, in addition to the approach taken to the compilation of data-driven macroprograms to node-level code. An implementation of the modular compilation framework is also discussed, as well as a graphical toolkit built around it that supports data-driven macroprogramming. Through experiments, it is shown that the code generated by the compiler matches hand-generated implementations of the applications, while drastically reducing the time and effort involved in developing real-world WSN applications.

26.1 Introduction

Wireless sensor networks (WSNs) enable low-cost, dense monitoring of the physical environment through collaborative computation, and communication in a network of autonomous sensor nodes, and are an area of active research [7]. Owing to the work done on system-level services such as energy-efficient medium access [25] and data-propagation [44] techniques, sensor networks are being deployed in the real world, with an accompanied increase in network sizes, amount of data handled, and the variety of applications [19, 26, 28, 45]. The early networked sensor systems were programmed by the scientists who designed their hardware, much like the early computers. However, the intended developer of sensor network applications is not the computer scientist, but the designer of the system using the sensor networks which might be deployed in a building or a highway. Throughout this chapter, the

A. Pathak (✉)
INRIA Paris-Rocquencourt, Rocquencourt, France
e-mail: animesh.pathak@inria.fr

term *domain expert* will be used to mean the class of individuals most likely to use WSNs—people who may have basic programming skills but lack the training required to program distributed systems. Examples of domain experts include architects, civil and environmental engineers, traffic system engineers, medical system designers. We believe that the wide acceptance of networked sensing is dependent on the ease of use experienced by the domain expert in developing applications on them.

The various approaches of application development currently available to the domain expert are discussed next.

26.1.1 Node-Level Programming

Since their early days, WSNs have been viewed as a special class of distributed systems and have been approached as such from an application development perspective as well. Consequently, application developers have thus far specified their applications at the level of the individual node where they use a language such as nesC [16], galsC, or Java to write the program, directly interacting with the node-level services stated earlier or a middleware [14, 36, 49] that aids in the programming process. The developer can read the values from local sensing interfaces, maintain application level state in the local memory, send messages to other nodes addressed by node ID or location, and process incoming messages from other nodes. However, in all these approaches, the application developer is responsible for ensuring that these individual finite state machines executing on the individual nodes of the WSN will interact to produce the desired result.

Owing to the large size and heterogeneity of the systems involved, as well as the limited distributed programming expertize of the domain experts, the above paradigm of *node-level programming* is not easy to use for sensor networks. This is believed to be a large obstacle holding back the wide acceptance of WSNs. For example, to develop an environment management application in nesC, a commonly used language in WSNs, the developer has to specify the functions at each node in terms of the respective components—one each for sensing the environment, communicating with other nodes, as well as controlling the actuators attached to each nodes.

One of the earliest toolkits proposed to reduce the programming effort was the sensor network application construction kit (SNACK) [20], which provides a component composition environment that allows developers to define explicit configurable parameters for application-level components. The SNACK user develops applications at the node level using a text-based description of *wiring* between components, several of which are libraries provided by the authors. These programs are analyzed by the compiler to generate maximally shared nesC expansions, which then have to be deployed just like normal nesC applications. The Flask language [33] facilitates node-level programming using data-flow graphs and provides facilities for composing atomic subgraphs across the network using a *flow* communication model. The application is specified in a variant of OCaml, and the behavior of individual processing elements is specified in nesC. The Flask compiler then generates node-level nesC code from the datagraph. In addition to the above, some graphical toolkits have also been proposed for WSN application development. The authors of Viptos [11] allow developers to model and simulate TinyOS applications in a graphical manner. A similar functionality is provided by GRATIS [18] where developers can use GME for easy modeling of TinyOS applications.

26.1.2 High-Level Abstractions for WSNs

In spite of the tools available for easing node-level application development, the application developer is still responsible for ensuring that the distributed application that results from these communicating node-level programs performs the necessary functions as desired and is also efficient in terms of the energy spent during its operations. Several techniques have been proposed to provide a higher level view of the sensor network. TinyDB [32] and Cougar [53] were the first works to abstract the sensor field as a database, allowing the user to execute queries over the sensed data in an SQL-like manner. The Task [9] tool kit makes designing and deploying TinyDB query-based application easy, where users can query the sensor data using SQL-like queries and also provide a visualizer for monitoring the network health and sensor readings. SenQ [52] enables user-driven and peer-to-peer in-network query issue by wearable interfaces and other resource-constrained devices. Complex virtual sensors and user-created streams can be dynamically discovered and shared, and SenQ is extensible to new sensors and processing algorithms.

Semantic streams [50] present each user with a 3-D rendering of the sensors in the test bed as well as all predicates that are queryable. The work in [51] builds on it by providing a spreadsheet approach to programming and managing data-querying applications in WSNs. In semantic middleware [6], applications are represented in a graphical interface as composable data sources and inference units which can be connected to retrieve required data by composition engines. jWebDust [10] provides a multi-tier application environment, where different sensor networks can be visualized as one to query the sensed data in a user-friendly manner.

In addition to database-like abstractions, another active area of resarch for WSNs has been abstractions for the *target-tracking* applications. EnviroSuite [30] is an object-based programming system that introduces the *environmentally immersive paradigm*. Its abstractions revolve directly around elements of the environment as opposed to sensor network constructs, such as regions, neighborhoods, or sensor groups. Object instances float across the network following (geographically) the elements they represent. The EnviroSuite Compiler (EIPLC) takes EnviroSuite code as input and outputs desired environmental monitoring applications in nesC, which then can be compiled by a standard nesC compiler and uploaded to the motes. The recent EnviroMic [31] application focuses on a distributed acoustic monitoring, storage, and trace retrieval system designed for disconnected operation.

Moving beyond domain-specific application of system-level thinking, there have been attempts to provide more general-purpose high-level application design for sensor networks. The work on SensorWare [8] and State-centric programming [29] provided some of the initial thoughts on the matter. This was followed by increased research in the field of sensor network *macroprogramming*, which aims to aid the wide adoption of networked sensing by providing the domain expert the ability to specify their applications at a high level of abstraction. In macroprogramming, abstractions are provided to specify the high-level collaborative behavior at the *system level*, while intentionally hiding most of the low-level details concerning state maintenance or message passing from the programmer.

Kairos [21] (and later, Pleiades [27]) is an imperative, control-driven macroprogramming language where the application designer can write a single program in a Python-like language with additional keywords to express parallelism. A 'centralized' program describes the activities at all nodes in the system and is translated into node-level binaries by a dedicated compiler.

Regiment [40] is a functional programming language, with support for region-based functions like filtering, aggregation, and function mapping. The Regiment primitives operate on a model of the sensor network as a set of continuous data streams. In [38], the authors introduced the TML intermediate language to represent the actions being performed at individual nodes. Regiment programs can be seen as *data-flow graphs*, with primitives such as *afold* combining functions and data on actual nodes to produce data. The work in [39] extends this to the *WaveScript* language which addresses applications working on live data streams.

In their work on COSMOS [1], the authors have presented the *mPL* macroprogramming language and the *mOS* operating system which can be used to program WSNs by way of task graphs, as long as all nodes of a single type have the same set of tasks running on them. Finally, MacroLab [22] (now supported by MacroDebugging [46]) provides a Matlab-like interface to WSN application developers, so that they can use operations such as `addition`, `max`, and `find` on sensor data addressed as *macrovectors*. This paradigm focuses on accessing and operating on data presented in matrix form, sometimes using different implementations (centralized versus distributed) of the same operation (e.g., `max`). The work in ATaG [3] is focused on *data-driven macroprogramming* (discussed in detail in Sect. 26.2), which allows the developer to specify the functionality of their application in terms of *tasks* that interact with each other only using the *data items* that they produce and consume.

For a detailed discussion of the various techniques available for application development on sensor networks, the reader is recommended to read the survey by Mottola et al. [37].

26.1.3 Macroprogram Compilation

In the context of macroprogramming for WSNs, we define *compilation* as the *semantics-preserving transformation of a high-level application specification into a distributed software system collaboratively hosted by the individual nodes*. The macroprogram compilation process has several challenges (highlighted in [42]), which must be addressed by the designers of compilation frameworks for macro-programming languages. The process of semantics-preserving transformation itself involves addressing challenges of correct and efficient conversion of representation. In addition, developers should be given the ability to express performance goals for the deployed system (e.g., in terms of expected network lifetime or latency) that the compiler should consider in optimizing the configuration of individual nodes and the allocation of different functionality to them.

As illustrated in Fig. 26.1, the ease of design provided by macroprogramming comes at a cost when compared to traditional node-centric programming. In the former approach, application developers reason at a high level of abstraction, while the process of converting the high-level representation to that of the individual nodes is delegated to a *compiler*. The higher the level of abstraction, the more work needs to be done by the compiler. This makes the process of generating the final running code significantly different from one solved by the node-level compilers currently seen in WSNs.

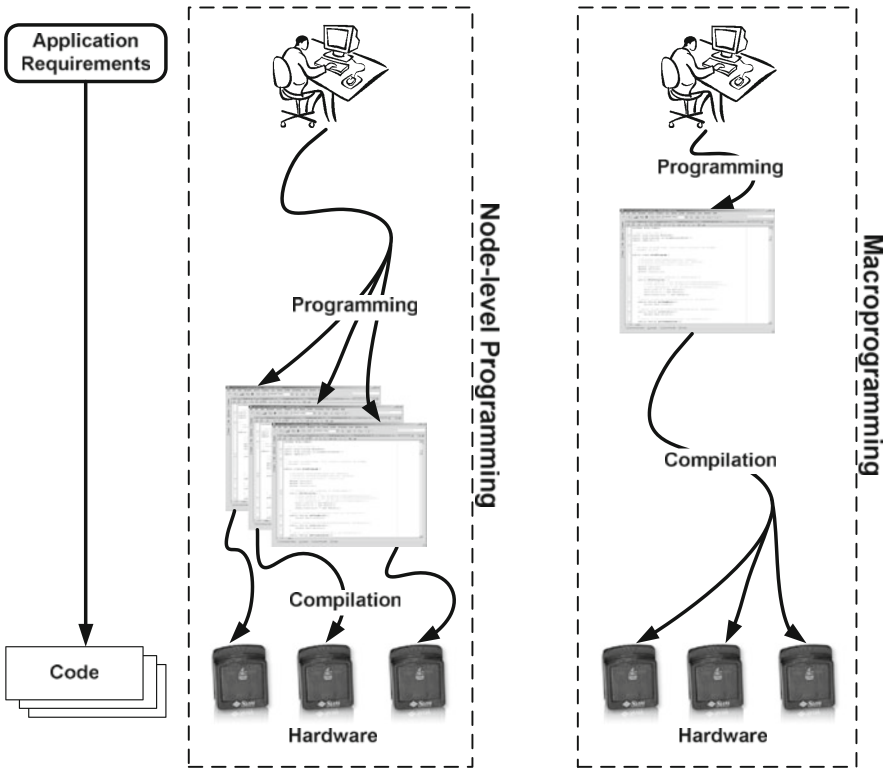


Fig. 26.1 Comparing node-centric and macro-programming

Data-driven macroprogramming languages allow the developer to specify the functionality of their application in terms of *tasks* that interact with each other only using the *data items* that they produce and consume. The focus of the rest of this chapter is to discuss in detail the design, implementation, and evaluation of a compilation framework to support a data-driven macroprogramming model called the *Abstract Task Graph* (ATaG) [3], whose salient features are described in Sect. 26.2. Overall, this chapter discusses the following:

- A general framework for compilation data-driven macroprogramming languages like ATaG: An overview of the compilation process is given in Sect. 26.3. The framework breaks down the process of converting the high-level specification to node-level functionality into a set of independent, isolated procedures—such as optimizing the placement of functionality on the real nodes or predicting communication costs. These different stages are connected through well-defined interfaces that allow for plugging in different modules implementing the various steps of compilation. The compilation framework is described in detail in Sect. 26.4.
- A demonstration of the flexibility and generality of the above framework by describing an end-to-end solution for compiling ATaG macroprograms. The proof-of-concept compiler, obtained by instantiating the different modules in the framework, provides the code to be deployed on each node, as well as an estimate of the message passing costs of the same. Moreover, the resulting code can be deployed on real-world nodes as well as in a simulation environment.
- Section 26.5 presents the design and implementation of *Srijan*—a graphical toolkit for WSN application development.
- Section 26.6 shows results from developing two realistic applications—building environment management (HVAC) [15] and highway traffic management [24]. The functionality of the compiler is assessed by inspecting and comparing the auto-generated code against a manually developed version of the same. The experiments show that using *Srijan*, application developers can specify and deploy their applications in a timely fashion, while having to write $\sim 2\%$ of total system code (or $< 10\%$ of application-specific code).

The details of the data-driven programming model of ATaG are discussed next.

26.2 Data-Driven Macroprogramming

As discussed in the previous section, macroprogramming of WSNs is an active area of research, with several programming paradigms currently being investigated. In this chapter, we focus on the *data-driven macroprogramming* paradigm, where the developers break up the functionality of their application into of *tasks* that interact with each other only using the *data items* that they produce and consume and do not share any state otherwise. This technique is shown to be especially useful in specifying a wide range of sense-and-respond applications [41]. The specific data-driven macroprogramming that we focus on here is called the *Abstract Task Graph*

(ATaG) [3]. ATaG includes an extensible, high-level programming model to specify the application behavior and a corresponding node-level runtime support, the data-driven ATaG runtime (DART) [2]. The compilation of ATaG programs consists of mapping the high-level ATaG abstractions to the functionality provided by DART. We now provide some background on these topics, as they represent the inputs and outputs of the transformation process, respectively.

26.2.1 Programming Model

ATaG provides a data-driven programming model and a mixed *imperative–declarative* program specification. A *data-driven* model provides natural abstractions for specifying reactive behaviors, while *declarative specifications* are used to express the placement of processing locations and the patterns of interactions.

The declarative portion of an ATaG program—a task graph—consists of the following components (see Fig. 26.2 for details).

- *Abstract Data Items*: The main currency of information in an ATaG program. They represent the information in its various stages of processing inside a WSN.
- *Abstract Tasks*: These represent the processing performed on the abstract data items in the system. Tasks do not share state with other tasks and can communicate only by producing and consuming data items. Tasks are annotated with *instantiation rules*, specifying where they can be located, as well as *firing rules*, specifying whether a task is triggered periodically or due to the production of certain data item(s).
- *Abstract Channels*: These connect tasks to the data items consumed or produced by them and are annotated with logical scopes [34], which express the interest of a task in a data item. In an ATaG program, a data item can only be produced by one abstract task, but can be consumed by many.

For each ATaG task, the developer also specifies the actions taken by the task using imperative code such as C or Java. Note that this code is concerned mostly with the processing of the data that the task has received and generating the data

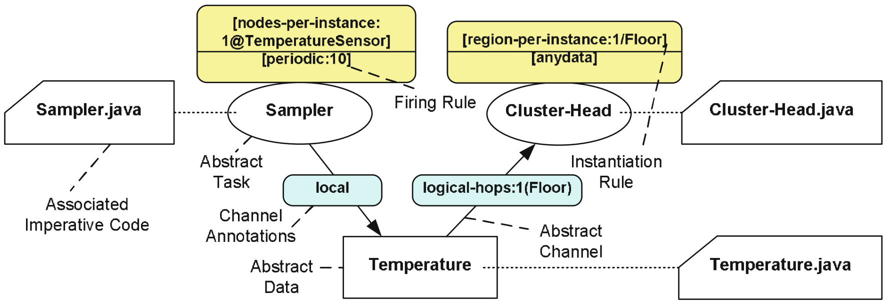


Fig. 26.2 ATaG program for data-gathering

items that the task will produce. To interact with the underlying runtime system, each task must implement a `handleDataItemReceived()` method for each type of data item that it is supposed to process. The task can output its data by calling the `putData()` method implemented by the underlying runtime system. Additionally, the developer needs to specify the details of each data item using imperative code.

Figure 26.2 illustrates an example ATaG program specifying a data gathering application [12] for building environment monitoring. Sensors within a cluster take periodic temperature readings, which are then collected by the corresponding cluster head. The *Sampler* task represents the sensing in this application, while the *Cluster-Head* task takes care of the collection. The *Temperature* data item is connected to both tasks using abstract channels. The *Sampler* is triggered every 10 s according to the `periodic` firing rule. The `any-data` rule requires *Cluster-Head* to run when a data item is ready to be consumed on *any* of its incoming channels. The `nodes-per-instance:`*q*`@Device` instantiation rule requires the task to be instantiated once every *q* nodes equipped with a specific device. According to `@TemperatureSensor`, the *Sampler* task in the example will be instantiated on every node equipped with a temperature device. Since the programmer requires a single *Cluster-Head* to be instantiated on every floor in the building, the `partition-per-instance:1/Floor` instantiation rule is used for this task. Its semantics is to derive a system partitioning based on the values of the node attribute provided (`Floor`). In this case, the programmer requires only *one* task to be instantiated in each partition.

As discussed earlier, the channels in the example program are annotated to express the interest of the producer and consumer tasks. The *Sampler* task generates data items of type *Temperature* kept `local` to the node where they have been generated. The *Cluster-Head* collects data not only from its own partition (floor), but also from adjacent ones. The `logical-hops:1(Floor)` annotation specifies a number of hops counted in terms of how many system partitions can be crossed, independent of the physical connectivity. Since *Temperature* data items are to be used within *one* partition (floor) from where they generated, they will be delivered to cluster heads running on the same floor as the task that produced them, as well as adjacent floors.

26.2.2 Runtime System

The node-level code output by the ATaG compiler is designed to run atop a supporting runtime hiding the underlying, platform-specific details. Figure 26.3 depicts the architecture of the data-driven ATaG runtime (DART) [2]. The functionality is divided into a set of modules to facilitate customization to various deployments.

The *ATaGManager* stores the declarative portion of the user-specified ATaG program that is relevant to the particular node. This information includes task annotations such as firing rule and I/O dependencies and the annotations of input and output channels associated with the data items that are produced or consumed by tasks on the node. The *DataPool* is responsible for managing all instances of abstract

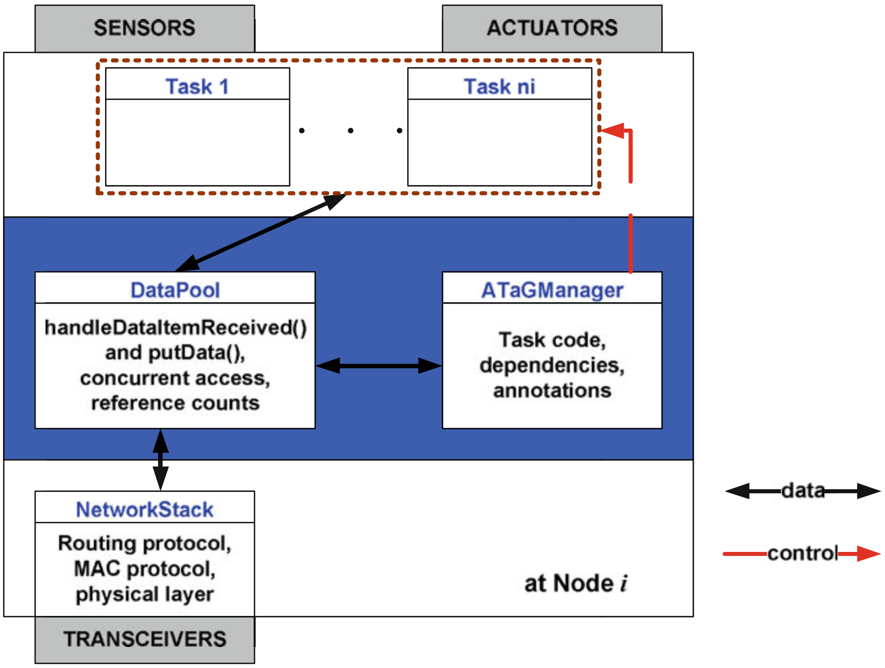

SENSORS
ACTUATORS
Task 1
Task ni
DataPool
handleDataItemReceived() and putData(), concurrent access, reference counts
ATaGManager
Task code, dependencies, annotations
NetworkStack
Routing protocol, MAC protocol, physical layer
TRANSCEIVERS
at Node *i*
data
control

Fig. 26.3 DART: Data-driven ATaG runtime system

data items produced or consumed at the node. The *NetworkStack* module is in charge of delivering data across nodes. The routing layer in it provides data-delivery across *logical scopes* [35, 36] by implementing a dedicated routing scheme. In particular, the inputs to this module include the data items and the *scope specifications* those are addressed to. A scope identifies, in a logical manner, the nodes an item is addressed to by referring to the relevant node attributes. For instance, a scope may specify all the nodes running the *Cluster-Head* tasks deployed on the first *Floor* as intended recipients. Other subsystems of the *NetworkStack* are in charge of communication with other nodes in the network and managing the physical layer protocols. Note that by itself, ATaG does not deal with fault tolerance. However, the runtime system and compiler developers are free to provide the user with an implementation that takes desired fault-tolerance requirements and suppors them by techniques such as task migration.

26.3 Compilation Process

In the previous section, we described the ATaG data-driven macroprogramming paradigm. In this section, we provide a formal definition of the process of compiling data-driven macroprograms to node-level code using the application given in Fig. 26.2 as example.

26.3.1 Input

The input to the compilation process consists of the following three components.
Abstract task graph (declarative part): Formally, an abstract task graph $A(AT, AD, AC)$ consists of a set AT of abstract tasks and a set AD of abstract data items. The set of abstract channels AC can be divided into two subsets — the set of *output channels* $AOC \subseteq AT \times AD$ and a set of *input channels* $AIC \subseteq AD \times AT$. In our example, the *Sampler* is AT_1 and *Cluster-Head* is AT_2, while *Temperature* is AD_1. AOC is $\{AT_1 \rightarrow AD_1\}$ and AIC is $\{AD_1 \rightarrow AT_2\}$.
Imperative code for each task: For each task and data item, the developer provides imperative code, which describes the actions taken at the host node when a task fires, and the internal details of the data item.
Network description: For every node in the target network N, the compiler is also given the following information:

- j: its unique ID.
- S_j: the list of sensors attached to j.
- A_j: the list of actuators attached to j.
- R_j: a set of $(RegionLabel, RegionID)$ attribute-value pairs to denote its membership in the regions of the network (e.g., {(Floor, 5), (Room, 2)}).

Runtime library files: These files contain the code for the basic modules of the runtime system that are not changed during compilation, including routing protocols.

26.3.2 Output

The goal of the compilation process is to generate a distributed application for the target network description commiserate with what the developer specified in the ATaG program. The output consists of the following parts:
Task assignments: The compiler must decide on the mapping to allocate the instantiated copies of the abstract tasks in AT to the nodes in N so as to satisfy all placement constraints specified by the developer.
Customized runtime modules: The compiler must customize the *DataPool* of each node to contain a list of the data items produced or consumed by the tasks hosted by it. It also needs to configure the *ATaGManager* module with a list of composed channel annotations, so when a data item is produced, the runtime can compute the constraints imposed on the nodes which are hosting the recipient tasks for it.
Cost estimates: The compiler also provides an estimate of the running cost of the application on the target deployment to provide feedback to the application developer. Note that the actual nature of the cost estimates returned can vary depending on the developer's needs. The costs returned may simply represent a measure of the communication overhead involved, e.g., in terms of messages exchanged per minute on a system-wide scale. Alternatively, finer-grained information may be computed, such as the expected per-node lifetime.

26.3.3 Process Overview

The *abstract* nature of the task graph is precisely what provides the application developer the desired high-level of abstraction needed to easily develop large and complex sense-and-respond applications for networked sensing systems. However, converting this high-level specification to a distributed application while preserving program semantics can be quite challenging. In this approach to the data-driven macroprogram compilation problem, the following major steps are envisaged.
Composition of abstract channels: Owing to ATaG's purely data-driven programming model, the developer only specifies relations between tasks and the data items they are producing (via AOC) and consuming (via AIC). While this provides a clean model to the application developer, traditional task allocation techniques work on task graphs with *direct dependency* links between tasks. To address the problem of generating such task graphs, each *path* $AT_i \rightarrow AD_k \rightarrow AT_j$ in the abstract task graph is converted to an *edge* $AT_i \rightarrow AT_j$.
Instantiating abstract tasks: The annotations of the abstract tasks in AT allow the developer to design one macroprogram for a variety of deployments. For example, the developer does not need to worry about the number of floors in the building, because he can use the `region-per-instance:1/Floor` instantiation rule. After the channels are composed, the compiler has the responsibility of expand this compact representation to the tasks into a full-fledged task graph that truly represents the data processing happening in the system.

The *instantiated task graph* (ITaG) is the internal representation used for this stage of the compilation process. It consists of multiple copies of each abstract task specified in the ATaG program, each ready to be assigned to individual nodes. The (directed) edges of the ITaG connect each task to the tasks that depend on it, i.e., the tasks that (a) copies of abstract tasks that consume the data item produced by it and (b) belong to the logical scope specified by the constraints in the connecting composed channel. Formally, the ITaG $I(IT,\ IC)$ is a graph whose vertices are in a set IT of instantiated tasks and whose edges are from the set IC of instantiated channels. For each task AT_i in the abstract task graph from which I is instantiated, there are $f(AT_i, N)$ elements in IT, where f maps the abstract task to the number of times it is instantiated in N. $IC \subseteq IT \times IT$ connects the instantiated version of the tasks. The ITaG I can also be represented as a graph $G(V,\ E)$, where $V = IT$ and $E = IC$. Additionally, each IT_j in the ITaG has a label indicating the subset of nodes in N it is to be deployed on. This overlay of communicating tasks over the target deployment enables the use of modified versions of classical techniques meant for analyzing task graphs.

For example, for the application in Fig. 26.4, since there are seven nodes with attached temperature sensors, $f(AT_1,\ N) = 7$, following the `1@Temperature Sensor` instantiation rule of the *Sampler* task. Similarly, $f(AT_2,\ N) = 3$, since the *Cluster-Head* task is to be instantiated once on each of the three floors. The figure shows one allocation of the tasks in IT, with arrows representing the instantiated channels in IC (it shows channels leading to only one instance of AT_2 for clarity). Note that although the ITaG notation captures the information stored in the abstract task graph (including the instantiation rules of the tasks and the scopes of

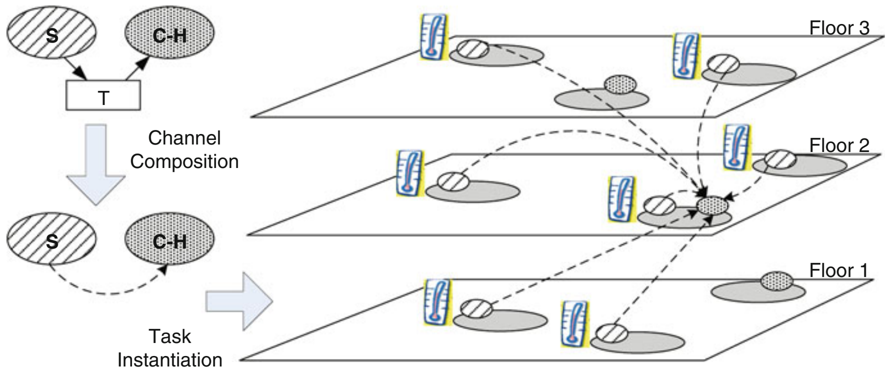


Fig. 26.4 An example illustrating the compilation process of our sample program

the connecting channels) it does not capture the *firing rules* associated with each task. The compiler's task involves incorporating the firing rule information while making decisions about allocating the tasks on the nodes.
Task mapping: This task graph with composed channels is then instantiated on the given target network. Figure 26.4 illustrates an example of a target network. The nodes are on three different floors and those marked with a thermometer have temperature sensors attached to them. In this stage, the compiler computes the mapping $M : IT \rightarrow N$, while satisfying the placement constraints on the tasks.
Customization of runtime modules: Based on the final mapping of tasks to nodes, and the composed channels, the *Datapool* and *ATaGManager* modules are configured for each node to handle the tasks and data items associated with it.

26.3.4 Challenges

The various stages described above each pose their own set of challenges. Since the channels in ATaG have logical scopes associated with them, the process of composing channels results in the (composed abstract channel) CAC_{ijk} being annotated with the union of *three* constraints. The first is that the node should have task AT_j assigned to it. The second(third) constraint is obtained by combining the instantiation rule of $AT_i(AT_j)$ with the annotation on the abstract channel connecting it to AD_k. For instance, in our example, after composition, AC_{121} is $\{$(*Cluster-Head is instantiated)* && (*Floor = Floor of Sampler or* $\pm 1)\}$. Depending on the complexity of scopes used in the channels, the resultant constraint can be further simplified by set operations to get a more compact constraint for the composed channel.

During the creation of the ITaG, maintaining the connections between the instantiated tasks in accordance with the placement rules on the tasks and the scope annotations on the channels is of utmost importance and can be time consuming if not done efficiently. Finally, an added complexity in the compilation process is brought by the large space of *optimizations* possible in the process to meet the user-specified performance goals (e.g., energy efficiency). Note that although tasks are assigned fixed locations at the end of the compilation process, *task migration* can happen later if the underlying system supports it. Even in such situations, a *good* initial task placement by a compiler using global knowledge can go a long way in creating efficient systems.

The following section describes how the components of the compilation framework act together to produce the outputs from the inputs, using the ITaG notation internally, and the implementation details of the ATaG compiler.

26.4 Compilation Framework

ATaG is designed to enable the addition of domain-specific constructs and customize the abstractions offered depending on the application requirements. This requires a flexible and extensible approach to the compilation problem. Ideally, the system designer should be given the ability to add new language constructs by implementing the required mappings without modifying any of the pre-existing compilation mechanisms. For instance, creating a new instantiation rule should not require modifications to the algorithms used to map tasks to nodes using an existing rule.

To address this issue, first the different steps involved in the compilation of ATaG programs were identified by factoring out orthogonal concerns and mechanisms. Next, considering the decomposition obtained, a modular compilation framework was designed, upon which the construction of the ATaG compiler was based. In this section, we describe the different modules of the framework (illustrated in Fig. 26.5), based on the problem definition of Sect. 26.3. The compilation stages are encapsulated in separated modules and defined generic interfaces between them so as to minimize inter-module dependencies. The modules are as follows:

Parser. The parser converts text files containing the declarative part of the program to an internal representation that is then used by the other modules. This process also involves a syntax check where errors such as duplicate task/data names and the existence of more than one producer task for one data item are identified and reported to the programmer.

In the current implementation, the declarative part of the ATaG program is specified using XML. This allows an easy integration of tools for the automated generation of XML specifications from graphical representations. The parser module is a simple XML parser that performs the aforementioned checks, assigns unique IDs to tasks and data items, and populates an internal data structure with the information.

Imperative code generator. Based on the parser output, the imperative code generator creates a set of files containing the basic declaration of the variables associated with each task and data items. The imperative part of the code provided by the programmer can then be plugged into these templates.

In the prototype implementation, the imperative part of an ATaG program is expressed using Java. As such, the current code generator creates Java files with unique numerical constants for each abstract task and data item corresponding to their ID. Then, it creates a separate class for each abstract task with basic functionality filled in (e.g., a thread instance with a loop for periodic tasks).

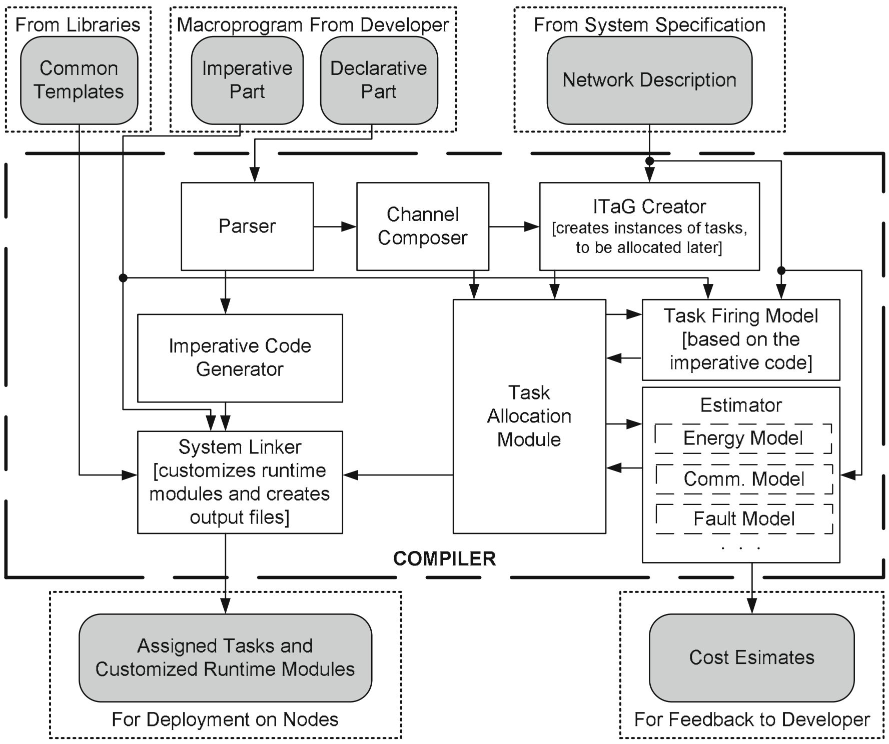

From Libraries
Common Templates
Macroprogram From Developer
Imperative Part
Declarative Part
From System Specification
Network Description
Parser
Channel Composer
ITaG Creator [creates instances of tasks, to be allocated later]
Imperative Code Generator
Task Allocation Module
Task Firing Model [based on the imperative code]
Estimator
Energy Model
Comm. Model
Fault Model
. . .
System Linker [customizes runtime modules and creates output files]
COMPILER
Assigned Tasks and Customized Runtime Modules
For Deployment on Nodes
Cost Esimates
For Feedback to Developer

Fig. 26.5 The ATaG Compilation Framework

Channel composer. Based on the declarative part of the ATaG program returned by the parser, this module performs the *composition* of channels to and from each data item to form edges of the ITaG, as described in Sect. 26.3.

Depending on the actual channel annotations supported, the prototype implementation may perform a range of operations, from a simple concatenation to complex operations that also consider the instantiation rules of the producer/consumer tasks.

ITaG creator. Based on the network description and the output of the channel translator, the ITaG creator first computes the number of distinct *target regions* for each task, i.e., the set of candidate nodes for hosting a given task. For instance, tasks instantiated with `nodes-per-instance:x` as instantiation rule have the entire system as target region. For tasks assigned by `partition-per-instance:x/PLabel`, each set of nodes with the same value for `PLabel` is a target region (e.g., each node in Floor 5). The ITaG creator then instantiates the required number of copies of each abstract task, attaching metadata to each instantiated task signifying its target region. The ITaG creator also computes the edges in this new graph, based on the composed channels. Note that, at this stage, tasks are instantiated but *not yet* assigned to nodes. That is done by the task allocator module, discussed next.

The implementation of this module performs the above operations using the network description read from a text file containing basic information on the nodes, e.g., their identifier, and set of attributes describing their characteristics, such as the sensing devices installed.

Task allocation module. As such, the allocation module is one of the most important parts of the compilation process, since it is responsible for computing a mapping from the set of instantiated tasks to the set of nodes. Note the task instantiation rules can be characterized as either *fixed* location (e.g., `nodes-per-instance:1`) or *variable* location (e.g., `nodes-per-instance:3`), depending on whether or not there is a unique way of instantiating the copies of a task given the network description. In this respect, an extremely large problem space exists depending on the annotations used, metrics to be optimized, and properties of the network. To perform its job, the allocation module relies on two further modules—the estimator and the task-firing model—described next.

In this implementation, this module performs task allocation in *two* passes. In the first pass, it assigns all the tasks with *fixed* locations. In the second pass, it assigns *variable* location tasks. For the latter, one of the initial implementations used a random task-assignment policy, with each node in the target region having an equal probability of hosting the instances of the task. However, due to the generality of the framework, more sophisticated mechanisms can be plugged in to achieve performance goals specified by the application designer. Several task-mapping algorithms for improved performance have also been proposed [43].

Estimator. Taking as inputs the network description and the task placement returned by the allocation module, the estimator computes the cost metric returned at the end of the compilation process. The framework gives great flexibility in instantiating this module, as its interface is designed to be generic w.r.t. the nature of information required. This allows application developers to explore the trade-off between the *quality* of the estimate obtained and the *time* required to obtain it. For instance, during the early design stages it is usually helpful to have a quick estimate of the communication costs, so that many alternative solutions can be explored. In this case, a simple but fast *estimation algorithm* can be employed that does not account for message losses. Conversely, when the application developer is to fine-tune the application, an actual simulation of the deployed application can be run within the estimator.

In the prototype system, both ends of the spectrum were implemented. On one hand, there is a naive estimator returning communication costs as if all the tasks produced data when fired and the underlying routing mechanisms were able to identify the optimal message routes. On the other hand, there is also a wrapper around SWANS/Jist [4]: a simulator able to run unmodified Java code on top of a simulated network. This plug-and-play capability highlights the power of the framework.

Task-firing model. It would appear that if one knew the exact paths taken by the data items, one can precisely estimate the cost of running a given task allocation. However, not all instantiated tasks produce data when they fire. For instance, although a *Temperature Sampler* task may produce a *Temperature* data item whenever it fires, an *Alarm* task may or may not produce an alarm depending on whether or not the temperature of the region is high enough. The task-firing model's function is to assign probabilities to the firing of various tasks in the program. Although this module is not mandatory for a working compiler, various approaches can be used to obtain the needed information — ranging from the developer providing profiling data obtained from previous runs of the system to static code analysis techniques [5, 13].

System linker. At the end of the whole process, the linker module combines the information generated by the various modules of the compiler into the code to be deployed on the nodes of the target system. More specifically, it configures the *ATaGManager* and *DataPool* modules in the node-level runtime depending on the task and data items handled at each node, and merges the imperative code provided by the application developer with the templates generated by the imperative code generator.

In the current implementation, the output of this module is a set of Java packages for each node. Note that these files are not binaries. They still need to be *compiled* in the classical sense, but that can be done by any node-level compiler designed for the target platform.

26.5 *Srijan*: Graphical Toolkit for Data-Driven WSN Macroprogramming

Since the goal of WSN macroprogramming research is to make application development easier for the *domain expert*, we believe that it is absolutely necessary to make *easy-to-use toolkits* for macroprogramming available to them in order to both make their task easier, as well as to gain feedback about the macroprogramming paradigms themselves. Although various efforts exist in literature for making WSN application development easier, very few general-purpose graphical toolkits for macroprogramming are publicly available for the application developer to choose from. In this section, we show how the macroprogram compilation framework discussed above has been incorporated into *Srijan* (named after the Sanskrit word for creation), an easy-to-use graphical front end to the various steps involved in developing an application using ATaG. Figure 26.6 shows the various components of this toolkit. The clear arrows show the *inputs*, while the gray arrows show the *output* of each component. The various components of *Srijan* are as follows.

Task graph description GUI

The ability of specifying a WSN application in a graphical manner as interconnected task and data items is a major part of ease of use provided by ATaG. In *Srijan*, the generic modeling environment (GME) [17] has been customized so that developers can easily specify their abstract task graphs, complete with channel and task annotations. Figure 26.7 shows how a developer can specify the task graph of a

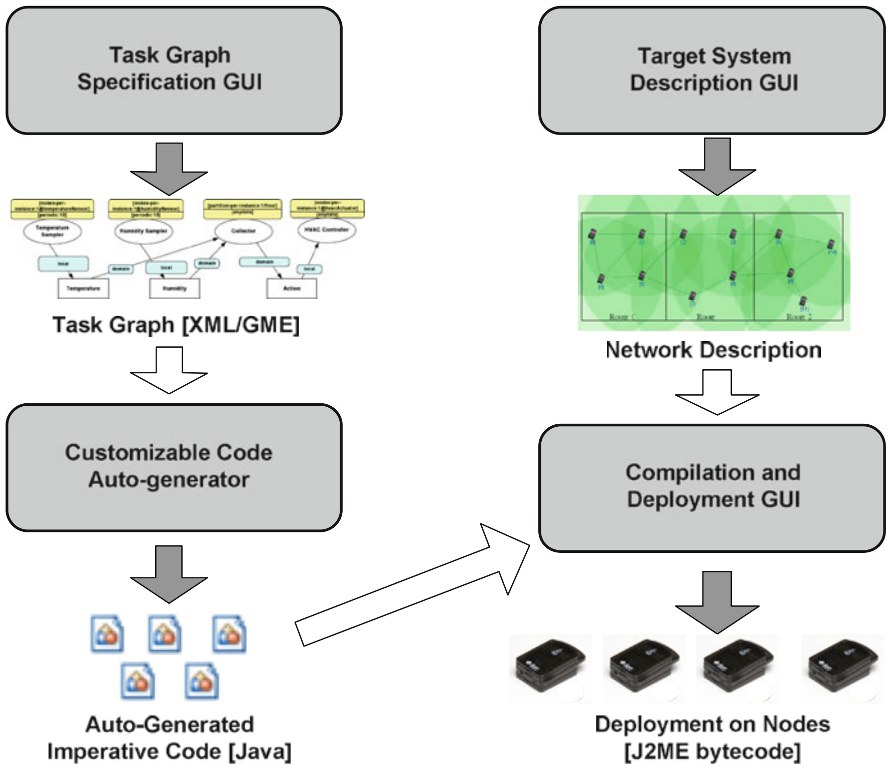

Task Graph Specification GUI
Task Graph [XML/GME]
Customizable Code Auto-generator
Auto-Generated Imperative Code [Java]
Target System Description GUI
Network Description
Compilation and Deployment GUI
Deployment on Nodes [J2ME bytecode]

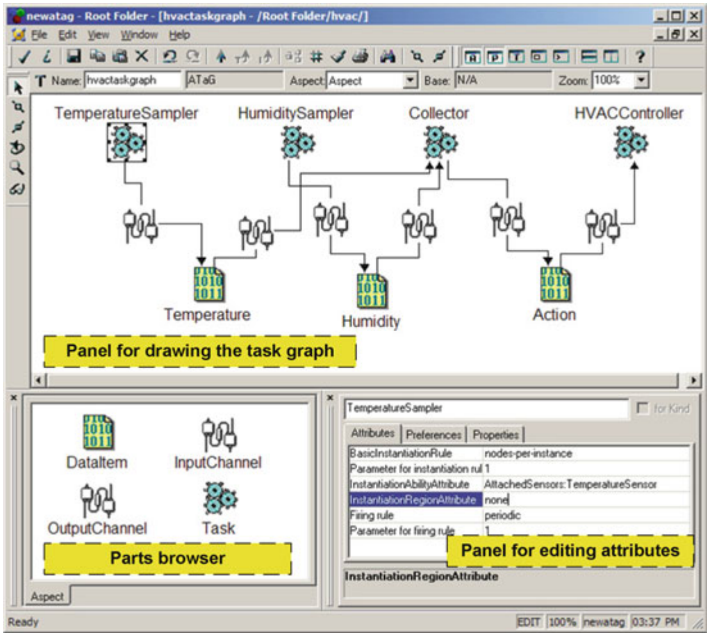
Fig. 26.6 Overview of application development using *Srijan*

building environment management application (detailed in Sect. 26.6.1) using the Srijan GUI. Once the details of the task graph are specified, *Srijan* generates an equivalent XML file, which can be used by the other modules.

Network description, compilation, and deployment GUI

The second part of *Srijan* (shown in Fig. 26.8) allows the developer to perform a set of actions. First, he can graphically specify the target network description, including the attributes of each node (alternatively, he can upload the specifications in a file). Second, the toolkit uses the task graph to generate a separate Java file for each task and data item with auto-generated communication and task-firing code.

Using the GUI, the developer can then invoke the compiler with the necessary parameters (optimization option, randomizer seed, etc.), which results in the generation of a set of Java files, one for each node in the target system, including the task and data code, as well as the customized runtime system modules. Finally, the developer can use *Srijan* to generate the bytecode for each node and deploy it to each Sun SPOT [48] over the air. The toolkit is currently under actively development and has been released for download [47].


newatag - Root Folder - [hvactaskgraph - /Root Folder/hvac/]
File Edit View Window Help
TemperatureSampler
HumiditySampler
Collector
HVACController
Temperature
Humidity
Action
Panel for drawing the task graph
DataItem
InputChannel
OutputChannel
Task
Parts browser
Aspect
Panel for editing attributes
Ready

Name: hvactaskgraph | ATaG | Aspect: Aspect | Base: N/A | Zoom: 100%

TemperatureSampler ☐ for Kind

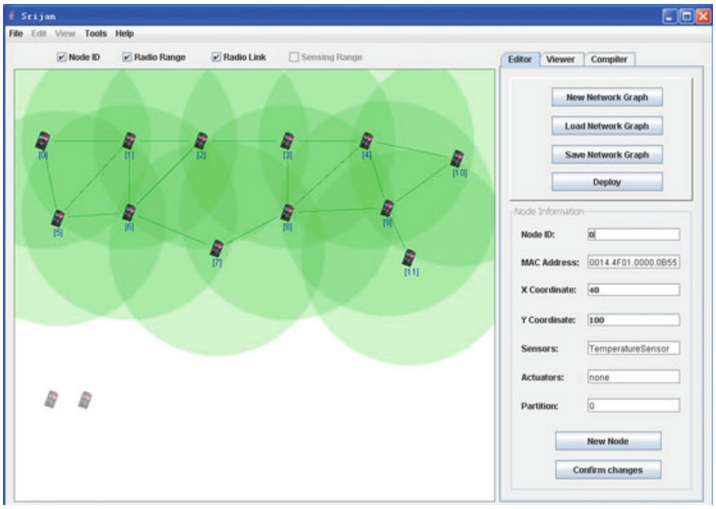
Attributes | Preferences | Properties

BasicInstantiationRule	nodes-per-instance
Parameter for instantiation rul	1
InstantiationAbilityAttribute	AttachedSensors:TemperatureSensor
InstantiationRegionAttribute	none
Firing rule	periodic
Parameter for firing rule	1

InstantiationRegionAttribute

EDIT | 100% | newatag | 03:37 PM

Fig. 26.7 Building environment management (HVAC) application in the Srijan task description GUI



☑ Node ID ☑ Radio Range ☑ Radio Link ☐ Sensing Range

Node Information

Node ID:	0
MAC Address:	0014.4F01.0000.0B55
X Coordinate:	40
Y Coordinate:	100
Sensors:	TemperatureSensor
Actuators:	none
Partition:	0

Fig. 26.8 Network description, compilation, and deployment using *Srijan*

26.6 Evaluation

26.6.1 Reference Applications

To demonstrate the effectiveness of the ATaG compiler, we consider two non-trivial applications and report on the *functionality* of the code generated, as well as the *performance* of the compilation process.

The first application, illustrated in Fig. 26.9, describes a highway traffic management system. In this case, two different sub-goals must be achieved — regulating the speed of vehicles on the highway by controlling speed limit displays and controlling the access to the highway by means of red/green signals on the ramps. The highway is divided into sectors, and sensors are deployed on the highway lanes and ramps to sense the speed and presence of vehicles, respectively. The sensed data goes through a multi-stage process where it is first aggregated w.r.t. a single sector to derive an average measure (*AvgSpeedCalculator* and *AvgQueueLength-Calculator* tasks) and then delivered to tasks deciding the actions taken in adjacent highway sectors (*SpeedLimitCalculator* and *RampSignalCalculator* tasks). The latter is expressed using the `logical-hops` construct relative to the `HighwaySector` attribute. Finally, data items describing the actions to perform are delivered to dedicated tasks instantiated on nodes equipped with the corresponding device, i.e., speed limit displays for the *SpeedLimitDisplayer* and ramp signals for the *RampSignalD-isplayer*.

The second application, depicted in Fig. 26.10, targets a *building environment management* system. Essentially, the processing is similar to the cluster-based data aggregation of Fig. 26.2, but now gathering data from two different types of sensors. The `@TemperatureSensor` and `@HumiditySensor` constructs are used to distinguish nodes with different types of sensing devices. Additionally, the cluster head also outputs data items representing actions to perform on the environment. These

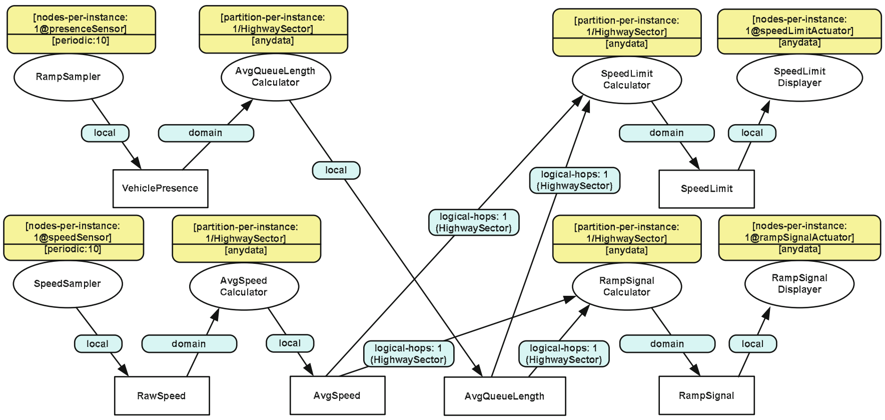

[nodes-per-instance: 1@presenceSensor] [periodic:10]
RampSampler
[partition-per-instance: 1/HighwaySector] [anydata]
AvgQueueLength Calculator
[partition-per-instance: 1/HighwaySector] [anydata]
SpeedLimit Calculator
[nodes-per-instance: 1@speedLimitActuator] [anydata]
SpeedLimit Displayer
local
domain
VehiclePresence
local
logical-hops: 1 (HighwaySector)
logical-hops: 1 (HighwaySector)
domain
local
SpeedLimit
[nodes-per-instance: 1@speedSensor] [periodic:10]
SpeedSampler
[partition-per-instance: 1/HighwaySector] [anydata]
AvgSpeed Calculator
[partition-per-instance: 1/HighwaySector] [anydata]
RampSignal Calculator
[nodes-per-instance: 1@rampSignalActuator] [anydata]
RampSignal Displayer
local
domain
local
logical-hops: 1 (HighwaySector)
logical-hops: 1 (HighwaySector)
domain
local
RawSpeed
AvgSpeed
AvgQueueLength
RampSignal

Fig. 26.9 An ATaG program for highway traffic management

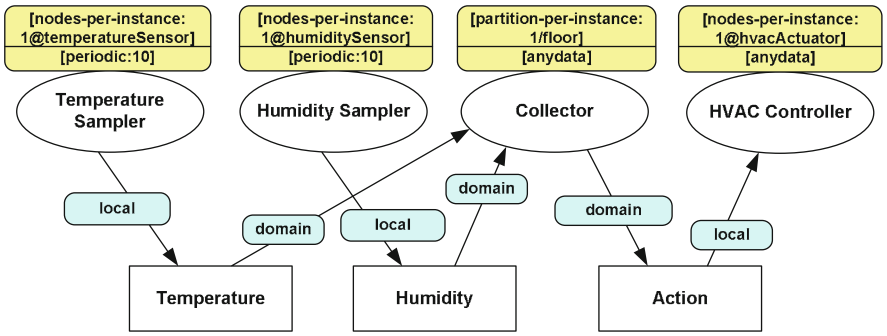


Fig. 26.10 An ATaG program for building environment management

items are input to an additional task that actually operates the heating, ventilation, and air conditioner (HVAC) devices in the building. As for this, the programmer requires the task to be instantiated on nodes with HVAC devices installed by means of the `@hvacActuator` construct.

26.6.2 Evaluation of the Compiler

Code functionality

The logic for both applications was hand coded to perform simulation studies on the underlying routing mechanisms [34]. The hand-written code also allowed to verify the functionality of the ATaG compiler, by comparing the automatically generated code with the one used in the aforementioned studies. Indeed, by comparing the simulation logs obtained using the SWANS/Jist [4] simulator, it was confirmed that the compiler-generated code is functionally equivalent to the hand-written version.

Settings for performance studies

Here we look at the *time* and *memory* taken to compile the above ATaG programs. Since the task-firing model used assumes that all tasks produce data when fired, the specific imperative code of the tasks does not influence the complexity of compilation. Rather, the compiler's performance is mainly dictated by the declarative part of an ATaG program and the characteristics of the deployment environment. More specifically, the following factors were seen to be pivotal in determining the time/memory taken to compile:

1. the number of abstract *tasks*, *data items*, and *channels*,
2. the nature of *instantiation rules* and *channel interests*, and
3. the *number of nodes* specified in the network description.

The complexity of the compilation task comes from different sources. The effort in composing channels is dependent on the actual channel annotations used, as well as the number of channels themselves. The ITaG creation stage becomes more complex as the complexity of the network grows. Note that this includes the number of logical regions the network can be divided into, as well as the variation in the attributes of the nodes. The size of the problem addressed by the task allocation module depends both on the network size as well as the constraints used in the program. For instance, placing a task whose instantiation rule is in the form `partition-per-instance:x/PLabel` requires more processing than placing a task with `nodes-per-instance:1`. All this in turn affects the performance of the system linker as it customizes the runtime on each node. Figure 26.11 reports the values of these factors seen in the sample applications.

In these tests, the compilation framework has been instantiated with the prototype implementations described in Sect. 26.4 for each module. In particular, the naive estimator and an *always-firing* task-firing model were employed. For each test performed, the compilation process was repeated 500 times to account for fluctuations due to concurrent processes.

Performance results

Figure 26.12 illustrates the performance of the compiler as a function of the number of target nodes. As expected, the time taken to compile an ATaG program grows quadratically as the number of nodes increases. This is due to the naive estimator used that computes the all-to-all shortest path with an algorithm whose time complexity is quadratic w.r.t. the number of vertices. However, fairly large instances can be compiled in reasonable time. For instance, slightly more than 10 s are needed to compile the traffic application for a target system with >250 nodes.

In addition, the memory consumed during the compilation process exhibits a linear increase with respect to the number of nodes in the deployed system. The source of this behavior is in the data structures employed in the ITaG creator and allocation modules that allocate a fixed amount of data for each target node. The memory consumed is always well within the limits of standard desktop PCs (<100 MB).

	Building	**Traffic**
Abstract Tasks	4	8
nodes-per-instance:x@PLabel	3	4
partition-per-instance:x/PLabel	1	4
Abstract Data Items	3	6
Abstract Channels	6	14
local	3	6
domain	3	4
logical-hops:1(PLabel)	0	4

Fig. 26.11 Complexity of the task graphs of sample applications

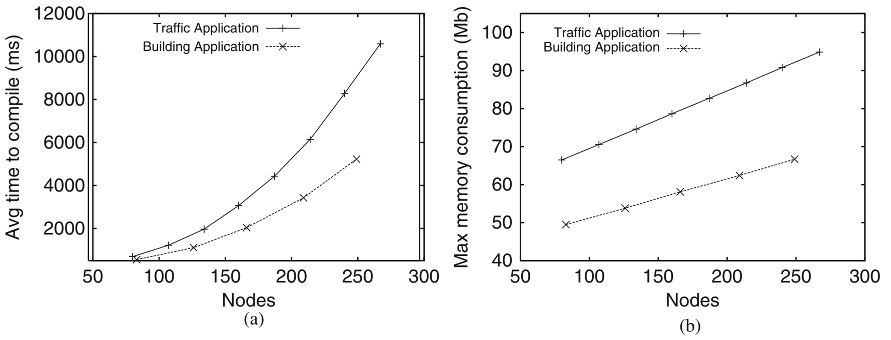


Fig. 26.12 Performance of the ATaG compiler w.r.t target network size. (**a**) Time taken to compile. (**b**) Maximum memory consumed during compilation

In exchange for the above costs in term of memory and time, the framework buys the developer *ease of use* in implementing the application using ATaG macro-programs, as discussed next.

26.6.3 Evaluation of the Toolkit

To evaluate the performance of *Srijan*, both the applications discussed in Section 26.6.1 were developed using it. For each of the applications, the complete end-to-end development was performed—starting from specifying the ATaG task graph to deployment of code on the nodes—using *Srijan*. The developer used a Pentium-4 2.8 GHz laptop with 1 GB of RAM running Windows XP for evaluation. The deployment was done onto the Sun SPOT [48] nodes, with a 180 MHz 32 bit ARM920T processor, 512K RAM, and 4 M flash memory. The nodes run the Squawk Java virtual machine directly out of flash memory and can run programs written using J2ME libraries. The Sun SPOT base station was used to deploy the code over the air (OTA) to the SPOTs. The Java hProf profiler [23] was used for measuring execution time.

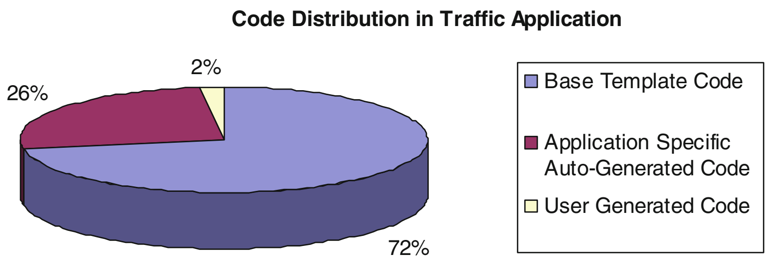
During the experiments, a variety of statistics was collected. The first metric was the time taken by the toolkit to (a) create the auto-generated imperative code templates, (b) allocate tasks to the nodes and generate per-node customized Java files, and (c) generate the Java bytecode for each node and deploy it over the air. In addition to the above times, the experiment also collected statistics regarding the amount of total code that was written by the application developer versus the code auto-generated by *Srijan*. Although the *line-of-code* metric is more a measure of the power of the ATaG compiler, the numbers are reported because (a) these numbers are of the J2ME-targeted implementation of the ATaG compilation framework and (b) this emphasizes the power of the ATaG macroprogramming paradigm which is made accessible to the application developer in a graphical manner by the *Srijan* toolkit.

	HVAC	**Traffic**
Imperative Code Gen. Time (ms)	1766	3422
Node-Specific Code Gen. Time (ms)	31967	77089
Per-node Deployment Time (s)	21	23
Source Files Edited by Developer	11	18
Total Number of Source Files	57	64
Lines of Application-specific Auto-generated Code	569	1019
Lines of Application-specific Code Written by Developer	60	81
Total Lines of Code	3433	3904
Task Graph Specification Time (min)	10	25
Imperative Code Editing Time (min)	17	60

Fig. 26.13 Costs involved in various stages of application development using *Srijan*

In addition to the above objective metrics, the experiment also measured the time it took for an application developer using *Srijan* to specify the ATaG task graph as well as the time taken in customizing the imperative code generated by it. Note that that these timings are variable from person to person, and more accurate statements can be made only after doing large user studies.

The experimental data is summarized in Fig. 26.13. Note that the time taken by *Srijan* to generate the files are within acceptable limits and are limited only by the hardware it is being run on, and in the case of deployment, also on the Java compiler used by the Sun SPOT SDK. More importantly, the developer had to write only a very small fraction of Java source files. The *total code* deployed



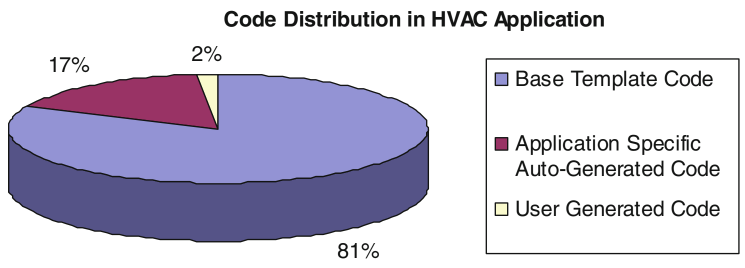


Fig. 26.14 Distribution of code generation effort

on each node consists of three components: (a) base template code—containing the DART libraries, (b) application-specific auto-generated code—generated by *Srijan*, and (c) user-generated code—written by the application developer to specify the details of the task and data. Figure 26.14 shows that the user-generated code is only around 2% of the total code. Even if the library code is neglected, *Srijan* generated $> 90\%$ of the application-specific code in each case. The importance of the time taken by the application developer in specifying the task graph and customizing the auto-generated code is highlighted by the fact that under normal circumstances, *Srijan* will be used by domain experts, e.g., civil engineers, who would have taken much more time customizing the runtime protocols and figuring out the task placements if it was not available as part of *Srijan*. These initial experiments demonstrate that the toolkit makes application development for WSNs more convenient for the domain expert.

26.7 Concluding Remarks

In spite of the developments in various areas of supporting applications on wireless sensor networks, the difficulty in application development still presents a hurdle to their wide acceptance. This chapter presented an overview of the various approaches available to WSN application developers for creating their applications, both at the node level and at the system level. Focussing specifically on data-driven macroprogramming, we discussed a general compilation framework for a data-driven macroprogramming language for sensor networks. The framework was then shown to be used for developing a compiler that can convert macroprograms written in ATaG into a running sensor system, which was followed by a discussion of *Srijan*—a graphical toolkit for end-to-end development of WSN applications using the ATaG data-driven macroprogramming language. Through experiments, it was shown that the time taken to compile the macroprogram depends closely on the complexity of both the macroprogram and that of the target sensor system, and also that using *Srijan*, developers can quickly develop realistic WSN applications while writing a very small fraction of the actual application code.

Although data-driven macroprogramming aims to help make WSN application design easy, it is up to the designers of the compiler and runtime system to make up for the loss of performance (e.g., high energy costs, shorter lifetimes) that inevitably accompany a rise in the level of abstraction by incorporating optimizations in the compilation process. Two such optimizations that can be looked at are placing tasks to reduce to communication costs and increase the system lifetime and performing logical-expression sharing when combining the scopes in the channel annotations of the task graph. We believe that future research in the domain will see experts in each stage of the compilation process developing better techniques and algorithms for their part.

Acknowledgments This work is partially supported by the National Science Foundation, USA, under grant number CCF-0430061 and CNS-0627028. The authors are grateful to Amol Bakshi, Luca Mottola, Gian-Pietro Picco, and Qunzhi Zhou for their assistance.

References

1. A. Awan, S. Jagannathan, and A. Grama. Macroprogramming heterogeneous sensor networks using cosmos. In: *EuroSys '07: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, ACM, New York, NY, pp 159–172, 2007. DOI http://doi.acm.org/10.1145/1272996.1273014.
2. A. Bakshi, A. Pathak, and V. K. Prasanna. System-level support for macroprogramming of networked sensing applications. In: *International Conference on Pervasive Systems and Computing (PSC)*, Las Vegas, Nevada, USA, 2005.
3. A. Bakshi, V. K. Prasanna, J. Reich, and D. Larner. The abstract task graph: A methodology for architecture-independent programming of networked sensor systems. In: *Workshop on End-to-End Sense-and-Respond Systems (EESR)*, Seattle, WA, USA, 2005.
4. R. Barr, Z. J. Haas, and R. van Renesse. Jist: An efficient approach to simulation using virtual machines. *Software Practice and Experience* 35(6):539–576, 2005.
5. G. Bernat, A. Burns, and A. Wellings. Portable worst-case execution time analysis using java byte code. In: *Proceedings of the 12nd Euromicro Conference on Real-Time Systems*, Maastricht, The Netherlands, 2000.
6. E. Bouillet, M. Feblowitz, Z. Liu, A. Ranganathan, A. Riabov, and F. Ye. A semantics-based middleware for utilizing heterogeneous sensor networks. In: *Proceedings of the 3rd International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Santa Fe, NM, USA, 2007.
7. A. Boukerche. *Algorithms and Protocols for Wireless and Mobile Systems*. Chapman & Hall/CRC, 2005.
8. A. Boulis, C. C. Han, and M. B. Srivastava. Design and implementation of a framework for efficient and programmable sensor networks. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*. ACM, New York, NY, pages 187–200, 2003. DOI http://doi.acm.org/10.1145/1066116.1066121.
9. P. Buonadonna, D. Gay, J. Hellerstein, W. Hong, and S. Madden. TASK: Sensor network in a box. In: *Second European Workshop on Wireless Sensor Networks, EWSN 2005*, Istanbul, Turkey, 2005.
10. I. Chatzigiannakis, G. Mylonas, and S. E. Nikoletseas. jWebDust: A java-based generic application environment for wireless sensor networks. In: *DCOSS*, Marina del Rey, California, USA, pages 376–386, 2005.
11. E. Cheong, E. A. Lee, and Y. Zhao. Joint modeling and design of wireless networks and sensor node software. Technical report, Electrical Engineering and Computer Sciences University of California at Berkeley, 2006.
12. W. Choi, P. Shah, and S. Das. A framework for energy-saving data gathering using two-phase clustering in wireless sensor networks. In: *Proceedings of the 1st International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS)*, Boston, MA, USA, 2004.
13. J. C. Corbett, M. B. Dwyer, J. Hatcliff, S. Laubach, C. S. Pasareanu, Robby, and H. Zheng. Bandera: Extracting finite-state models from java source code. In: *Proceedings of the 22nd International Conference on Software Engineering (ICSE)*, Limerick, Ireland, 2000.
14. C. Curino, M. Giani, M. Giorgetta, A. Giusti, A. L. Murphy, and G. P. Picco. Tinylime: Bridging mobile and sensor networks through middleware. In: *Third IEEE International Conference on Pervasive Computing and Communications, 2005. PerCom 2005*, pp 61–72, Kauai, Hawaii, USA, 2005.
15. M. Dermibas. Wireless sensor networks for monitoring of large public buildings. Technical report, University at Buffalo, 2005.
16. D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: A holistic approach to networked embedded systems. In: *Proceedings of Programming Language Design and Implementation (PLDI)*, San Diego, CA, USA, 2003.
17. The generic modeling environment, http://www.isis.vanderbilt.edu/projects/gme. Accessed on October 2010.

18. GRATIS: Graphical development environment for tinyos. http://www.isis.vanderbilt.edu/projects/nest/gratis/index.html. Accessed on October 2010.
19. Habitat monitoring on the Great Duck Island. www.greatduckisland.net
20. B. Greenstein, E. Kohler, and D. Estrin. A sensor network application construction kit (SNACK). In: *2nd ACM Conference on Embedded Networked Sensor Systems*, Baltimore, MD, USA, 2004.
21. R. Gummadi, O. Gnawali, and R. Govindan. Macro-programming wireless sensor networks using Kairos. In: *Proceedings of the 1st International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Marina del Rey, California, USA, 2005.
22. T. W. Hnat, T. I. Sookoor, P. Hooimeijer, W. Weimer, and K. Whitehouse. Macrolab: A vector-based macroprogramming framework for cyber-physical systems. In: *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, ACM, New York, NY, pages 225–238, 2008. DOI http://doi.acm.org/10.1145/1460412.1460435
23. HPROF: A heap/cpu profiling tool in J2SE 5.0. http://java.sun.com/developer/ technicalArticles/Programming/HPROF.html. Accessed on October 2010.
24. T. T. Hsieh. Using sensor networks for highway and traffic applications. *IEEE Potentials* 23(2):13–16, 2004.
25. V. Jain, R. Biswas, and D. P. Agrawal. Energy efficient and reliable medium access for wireless sensor networks. In: *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Helsinki, Finland, 2007.
26. B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In: *Proceedings of ACM/IEEE MobiCom*, Boston, MA, USA, 2000.
27. N. Kothari, R. Gummadi, T. Millstein, and R. Govindan. Reliable and efficient programming abstractions for wireless sensor networks. In: *PLDI '07: Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation*, ACM, New York, NY, pages 200–210, 2007. DOI http://doi.acm.org/10.1145/1250734.1250757.
28. B. Krishnamachari. *Networking wireless sensors*. Cambridge University Press, 2006.
29. J. Liu, J. Reich, and F. Zhao. State-centric programming for sensor-actuator network systems. *IEEE Pervasive Computing*, 2(4):50–62, 2003.
30. L. Luo, T. F. Abdelzaher, T. He, and J. A. Stankovic. Envirosuite: An environmentally immersive programming framework for sensor networks. *Transaction on Embedded Computing System* 5(3):543–576, 2006.
31. L. Luo, Q. Cao, C. Huang, L. Wang, T. F. Abdelzaher, J. A. Stankovic, and M. Ward. Design, implementation, and evaluation of enviromic: A storage-centric audio sensor network. *ACM Transaction on Sensor Network* 5(3):1–35, 2009. DOI http://doi.acm.org/10.1145/1525856.1525860.
32. S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: An acquisitional query processing system for sensor networks. *ACM Transaction on Database System* 30(1):122–173, 2005. DOI http://doi.acm.org/10.1145/1061318.1061322.
33. G. Mainland, M. Welsh, and G. Morrisett. Flask: A language for data-driven sensor network programs. In: Technical Report TR-13-06, Harvard University Technical Report, 2006.
34. L. Mottola, A. Pathak, A. Bakshi, V. K. Prasanna, and G. P. Picco. Enabling scoping in sensor network macroprogramming. In: *Fourth IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)* Pisa, Italy, 2007.
35. L. Mottola and G. P. Picco. Logical neighborhoods: A programming abstraction for wireless sensor networks. In: *Proceedings of the the 2nd International Conference on Distributed Computing on Sensor Systems (DCOSS)*, San Francisco, CA, USA, 2006.
36. L. Mottola and G. P. Picco. Programming wireless sensor networks with logical neighborhoods. In: *Proceedings of the 1st International Conference on Integrated Internet Ad hoc and Sensor Networks (InterSense)*, Nice, France, 2006.
37. L. Mottola and G. P. Picco. Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Computing Surveys* (accepted).

38. R. Newton, Arvind, and M. Welsh. Building up to macroprogramming: An intermediate language for sensor networks. In: *Proceedings of the 4th International Conference on Information Processing in Sensor Networks (IPSN)*, Los Angeles, CA, 2005.
39. R. Newton, L. Girod, M. Craig, G. Morrisett, and S. Madden. Design and evaluation of a compiler for embedded stream programs. In: *Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2008)*, Tucson, AZ, USA, 2008.
40. R. Newton and M. Welsh. Region streams: Functional macroprogramming for sensor networks. In: *Proceedings of the 1st International Workshop on Data Management for Sensor Networks (DMSN)*, Toronto, Canada, 2004.
41. A. Pathak, L. Mottola, A. Bakshi, V. K. Prasanna, and G. P. Picco. Expressing sensor network interaction patterns using data-driven macroprogramming. In: *Third IEEE InternationalWorkshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS 2007)*, White Plains, NY, 2007.
42. A. Pathak and V. K. Prasanna. Issues in designing a compilation framework for macroprogrammed networked sensor systems. In: *Proceedings of the the 1st International Conference on Integrated Internet Ad hoc and Sensor Networks (InterSense)*, Nice, France, 2006.
43. A. Pathak and V. K. Prasanna. Energy-efficient task mapping for data-driven sensor network macroprogramming. In: *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Santorini, Greece, 2008.
44. O. Powell, P. Leone, and J.D.P. Rolim. Energy optimal data propagation in wireless sensor networks. *Journal of Parallel and Distributed Computing* 67(3):302–317, 2007.
45. M. Rahimi, M. Hansen, W. Kaiser, G. Sukhatme, and D. Estrin. Adaptive sampling for environmental field estimation using robotic sensors. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Canada, 2005.
46. T. Sookoor, T. Hnat, P. Hooimeijer, W. Weimer, and K. Whitehouse. Macrodebugging: Global views of distributed program execution. In: *SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM, New York, NY, pages 141–154, 2009. DOI http://doi.acm.org/10.1145/1644038.1644053.
47. *Srijan*—Graphical WSN application development toolkit. https://gforge.inria.fr/projects/srijan/.
48. Sun$^{\text{TM}}$ Small Programmable Object Technology (Sun SPOT), www.sunspotworld.com.
49. K. Whitehouse, C. Sharp, E. Brewer, and D. Culler. Hood: A neighborhood abstraction for sensor networks. In: *Proceedings of the 2nd International Conference on Mobile systems, applications, and services (MOBISYS)*, Boston, MA, USA, 2004.
50. K. Whitehouse, F. Zhao, and J. Liu. Semantic streams: A framework for composable semantic interpretation of sensor data. In: *European Workshop on Wireless Sensor Networks (EWSN)*, Zurich, Switzerland, 2006.
51. A. Woo, S. Seth, T. Olson, J. Liu, and F. Zhao. A spreadsheet approach to programming and managing sensor networks. In: *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*. New York, NY (2006). DOI http://doi.acm.org/10.1145/1127777.1127842.
52. A. D. Wood, L. Selavo, and J. A. Stankovic. SenQ: An embedded query system for streaming data in heterogeneous interactive wireless sensor networks. In: Nikoletseas S.E., Chlebus B.S., Johnson D.B., and Krishnamachari B., editors. DCOSS, volume of *Lecture Notes in Computer Science*, Springer, Berlin, pages 531–543, 2008.
53. Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Record* 31(3):9–18, 2002. DOI http://doi.acm.org/10.1145/601858.601861.

Chapter 27
Toward Integrated Real-World Sensing Environment — Applications and Challenges

Srdjan Krco and Konrad Wrona

Abstract Growing popularity of wireless sensor network (WSN) applications and machine-to-machine (M2M) communication has recently led to emergence of a new class of network traffic. The communication between *things* introduces network traffic characteristics very different from traditional human-centric communication. As more WSN and M2M services are expected to use mobile network as a backhaul, it is important to understand the impact that their traffic will have on the radio access network. WCDMA radio access networks are dimensioned using traffic models containing traffic characteristics for a number of services offered by the operators and their networks today. In this chapter, the potential capacity impact of several selected sensor-based applications on the WCDMA radio access network, both dedicated and common channels, is presented. The results thus far have shown that currently 3G mobile networks can successfully cope with the new WSN traffic. With the increase of WSN traffic, optimizations of protocols and solutions will be required and radio bearer mapping can be a deciding factor in determining the size of the radio access network and the required resources.

27.1 Introduction

Over the last 10 years, wireless sensor networks have become an important research topic. Starting as a subfield of ad hoc networking research and then slowly differentiating into a research area on its own, wireless sensor networks (WSN) have covered the path from a purely academic research to a commercial reality.

The *ubiquitous computing* was originally defined by Mark Weiser [46] as disappearing technologies that *weave themselves into the fabric of everyday life until they are indistinguishable from it*. This vision of ubiquitous computing was based on a number of small sensors and actuators, machines in general, embedded in everyday items, tied together by a communication network, with a range of applications utilizing this infrastructure. With this vision that connects physical and digital worlds in mind, a plethora of applications and scenarios was envisaged ranging from military

S. Krco (✉)
Ericsson, V. Popovica 6, 11000 Belgrade, Serbia,
e-mail: srdjan.krco@ericsson.com

to health to environment monitoring. The early scenarios were mainly considering large sensor networks, with thousands or more homogeneous sensors deployed in an ad hoc manner over an area. The sensors would self-organize, automatically establish communication channels, discover routes to sink nodes, and collaboratively aggregate collected data before delivering it to a sink node. Usually, these networks were deployed with one particular task in mind, such as monitoring temperature, monitoring movement of people and vehicles, or monitoring health parameters of a person. The WSNs were considered as isolated and self-contained systems with a WSN gateway as the only node that interacts with the outer world, i.e., the users of the sensor information. This view has changed over the years toward deployment of not so large, heterogeneous sensor networks, considering the networks as sensor service providers and enabling their interaction over Internet, fixed and mobile, in order to create complex services and information to the end users [3, 13, 20].

In this chapter, we discuss two important tiers of research and development in wireless sensor networks. One of them is integration of WSN into civilian applications and, in particular, the influence of wide deployment of WSNs on the public mobile systems. The other one is integration of WSNs into military applications, which contrary to the civilian applications rely on dedicated and closed communication information systems, but introduce many specific security challenges.

In Sects. 27.4, 27.5, 27.6, and 27.7 we present the initial results of a WCDMA radio access network capacity calculation and analysis as affected by a few selected WSN-based applications. Impacts of the traffic generated by these applications on the radio network resources and configuration have been analyzed, in particular regarding the most efficient mapping of traffic to the radio access bearers (RAB) from both the application and the network point of view. The analysis was done based on the features and characteristics of the existing WCDMA networks.

27.2 Military Perspective

Military applications were one of the main drivers for early development of WSN technology. Over the years wireless sensor networks and RFID technology have constantly continued to gain importance in the military environment. Situational awareness and informational superiority are important elements of the modern military doctrine — the network-centric operations [5]. Both WSN and RFID are critical technologies for enabling this capability. The main objective of using WSN and RFID in military systems is to provide the so-called information superiority [4]. Information superiority derives from the ability to create a relative information advantage vis-a-vis an adversary.

The typical applications where use of WSN and RFID can offer information superiority include logistics, force protection, and combat assistance [47]. Example of current NATO projects related to remote sensing and tracking includes perimeter monitoring and convoy tracking in Afghanistan. DARPA has sponsored several projects related to use of sensor networks in military applications. Mobile mines equipped with wireless communication interface and a micro controller were designed to self-organize, form a mine field as well as to redeploy to cover potential gaps in the mine field coverage [32]. Large networks of sensors that are deployed by scattering the sensor nodes over a battlefield from an airplane were designed to monitor movement of people and vehicles [29]. Wireless sensor networks were also used to accurately detect the location of the sniper shooters [45].

Use of sensor networks and RFID technology in military environment introduces several challenges. In addition to high reliability and environmental resistance, the military systems, especially deployed in areas potentially controlled by an adversary, have to provide an adequate level of tamper resistance or tamper evidence. The communication links, especially wireless channels, have to be adequately secured, too. The system has to be easy to deploy, operate, maintain, and recover from possible faults and attacks. And, finally, the solution has to be easily integrated with the existing military systems.

One of the fundamental challenges in military communications and information systems (CIS) is dealing with different security classification levels of information. Most of the current military systems operate according to Bell-La Padula security model [8]. Extra care has to be taken when designing information flows within the system in order to meet the relevant organizational security policies. In particular, a flow of information from the *low* domain (e.g., unclassified network of sensors) to the *high* domain (e.g., secret network including intelligence analysis system and repository of historical data) requires enforcement of a one-way information flow, e.g., by using a data diode [41]. However, use of such one-way communication device implies also that no control information or queries can be sent back directly from the secret network to the sensors. Simple solutions, such as including sensor network itself in the secret domain, are not practical because of the cost and technical constraints.

Discussion of classification of information leads also to an interesting question concerning when the data collected by sensors becomes a classified information. As the raw data collected by sensor nodes can be in most cases sensed by anybody, including the enemy, it is often regarded as unclassified information. Therefore, both sensors and communication channels between and from the sensors do not need to meet the strict requirements on security of classified CIS, which would be difficult to meet due to the technological limitations. However, the aggregated or processed data is often regarded as classified, which implies that sink nodes or systems performing in-network processing might be regarded as classified, too.

Another interesting question is the definition of a term *sensor network*. Typical WSN scenarios studied in academia assume a large-scale distributed network of tiny autonomous devices. This challenging environment provides an interesting platform for identification and investigation of many new research issues and can lead to interesting practical solutions in the future. However, most of the currently deployed military sensing systems rely on much bigger sensors and much more centralized communication topologies. In fact, the term *sensor network* is often used in military environment in order to describe an interconnected system of radar stations and other long-range sensing instruments.

One of the important challenges that have to be addressed before the WSN gains widespread acceptance in military applications is that the currently proposed WSN architectures often do not provide adequate security. Wireless sensor networks face some usual security threats, including threats to confidentiality, integrity, and availability [19]. However, the constraint resources available to the nodes make addressing these threats a challenging task [26]. The spectrum of the technical security issues introduced by the classical WSN scenarios is extremely wide, ranging from physical security of the nodes through lightweight cryptographic mechanisms [11] and efficient security protocols to secure data aggregation [36] and secure integration into context-aware applications [10]. The highly distributed character of the WSN has also an important impact on choice of security mechanisms which are suitable for securing communication and data processing within the network. Interestingly enough, although the distributed character and the large scale of sensor networks may be an obstacle for using some well-known security mechanism, e.g., used in the Internet, at the same time these characteristics open opportunities for developing new approaches to security. The possible approaches include relying on redundancy of evidence and on limited ability of an attacker to gain control over substantial number of nodes or to monitor the substantial amount of communication exchange between nodes.

The security in WSN is an important R&D topic, which not only is critical for enabling real-life applications of sensor networks but also allows us to better understand limitations of existing security mechanism and fosters development of new, more efficient, alternatives. Despite substantial amount of research performed in this area in the recent years, many of the security problems are still open and provide interesting area for further investigation.

27.3 Civilian Perspective

The range of civilian WSN applications is huge. Sensors and actuators embedded into our environment can be of great benefit in numerous situations and everyday life. Environmental monitoring is one such domain. Monitoring of Great Barrier Reef [17] provides data about this remote and vast area ensuring continuous insights into the current status of sea life thus helping in protecting and preserving it. Forest fires have caused huge destructions globally, with Greece and Australia being recent examples. Deployment of a large number of wireless sensor nodes across endangered forests can help in identifying the fires at early stages thus facilitating rapid reaction of public services and consequently saving lives and property as well as minimizing forest destruction [16]. Air quality is a topic of great importance, especially in urban settings where air pollution caused by traffic and other sources of pollution is concentrated. In [33], a prototype air quality reporting system described tries to model the world from a sensor-centric view, with each observed value being associated with a geographic and temporal position, as well as the unit of measurement, data quality value, and process description.

The increasing cost of providing health care in medical institutions combined with the advancements in sensor and wireless technology has created a lot of interest in research and design of mobile and personal health monitoring systems, the so-called mHealth solutions [18, 21, 44]. Small sensors capable of monitoring various health parameters (ECG, temperature, pulse oximetry, blood pressure, blood sugar level, posture monitoring, etc.) communicate wirelessly (for example, using Bluetooth or other short-range wireless technology) with a mobile phone or a specialized device and transmit measured data to a central repository where it can be accessed by medical personnel over Internet [22, 25, 43]. In addition to the body-worn sensors, the systems often include sensors deployed in the user's home environment to monitor user's activity and movement. This is of particular interest for elderly monitoring applications to enable monitoring of the daily routine and execution of various tasks by the users in addition to the health parameters. The research addresses not only monitoring of users with health problems (for example, after hospitalization or with chronic health problems) but also supporting healthy people during workouts and training or specialized programs like weight loss to enable adaptation of the workout intensity to the overall health conditions as well as to provide motivation to users to endure throughout the training [6].

With the rapid proliferation of mobile networks and advancement of mobile devices, the list of interesting WSN application domains has been extended by the so-called *participatory sensing* that relies on people and their mobile devices (e.g., mobile phones) to gather information about the physical world [9, 14, 37]. Further on, actuators have been included into the networks of sensors [38] as well as static and mobile robots [40]. Machine-to-machine (M2M) communication as the way to connect individual machines and devices over Internet (and primarily over mobile networks) has been perceived by mobile network operators as a way to substantially increase the number of subscribers, create new services, and generate additional revenue utilizing the existing network infrastructure.

In the recent years, with the first commercial deployments, the focus of the WSN research has slowly started to shift toward integration of sensor networks with Internet and mashing up of information provided by these networks to create new service and added value to the users [13]. Opening up WSNs and making them a part of a wider system led to creation of wireless sensor network frameworks that aim at providing a set of standardized interfaces, protocols, and services to facilitate an efficient interaction with heterogenous WSN. The offered services range from discovering suitable WSNs and sensor nodes based on semantic queries through aggregating and processing observed data to dynamically creating new, virtual sensors by combining existing ones to provide information that otherwise would not be available [2, 15, 23, 35]. With these solutions in place, a tighter integration of WSNs with other networks and systems will be possible thus facilitating creation of a range of new services and supporting innovative concepts, such as smart cities, smart transport, smart health, and smart homes, and leading to the overall vision of the Internet of Things and, more general, the Future Internet. In fact, one of the most important aspects and envisioned innovations of the Future Internet is the support for inclusion of context information about the physical environment and making it available to all users in a similar manner the web services are available and used today. All these applications and technologies that enable interaction of the physical and the digital worlds and integration of the information about the physical world to Internet are referred to as *Internet of Things* or *Real World Internet* [42].

As the Internet of Things (IoT) is becoming a very important element of the modern communication systems, it is necessary to fully understand how the traffic generated by a plethora of IoT applications impacts the networks today and how to improve network protocols and procedures to ensure more efficient handling of new network users, applications, and services.

A significant number of WSNs and *things* will be connected to Internet via mobile networks. In comparison to the number of the *traditional* mobile network subscribers (currently around 4 billion subscribers in the world), wireless sensor network deployments are still low in numbers and are not currently seen as a significant source of additional traffic load for mobile networks. However, if the existing trends continue and the forecasts prove to be correct [12], we can expect that the number of WSNs, machines, and in general *things* using mobile networks will be growing rapidly and will eventually outnumber the traditional, human subscribers. To ensure optimum dimensioning of the networks and utilization of network resources, it is necessary to understand how this new class of users will use mobile networks and how will the traffic they generate impact the existing networks in terms of the capacity requirements, network configuration, quality of service, impact on the existing services, etc.

27.4 Selected WSN Applications and Traffic Models

WSN and M2M application space is broad and diverse as highlighted above. For the purpose of traffic modeling, we focus on the application domains that are already on the market or will most likely be available on the market very soon [7, 27]. From the WCDMA networks point of view, the most important applications are those that are expected to have high penetration levels and to use mobile networks to transfer data and interact with remote users. The important application parameters that affect mobile network include application requirements in relation to delay and service guarantees, predominant direction of data transfer of the application (downlink or uplink) data transfer, data volume per transaction, and frequency with which transactions are generated.

Based on these criteria and taking into account the available market forecasts for each of the domains [27], we selected a subset of applications that are expected to have the most significant impact on mobile networks in the coming years. The following application domains were included in our analysis: control and automation, transportation, environmental monitoring for emergency services, health care as well as frequent transfer of small amounts of data that is likely to be generated by WSN frameworks [13].

Within these domains, particular use cases were analyzed and their data delivery models included in the overall WSN applications traffic mix. Considering a wide range of WSN applications, it is natural that they have different expectations of mobile networks with regard to the quality of service provided. This makes it impossible to create one WSN traffic model as an input to the WCDMA radio access network dimensioning. Instead, each selected application was analyzed separately, categorized according to its service class, and mapped to the appropriate existing RABs. Although the study considers sampling periods on a sensor level, these are only used as references for determining a reasonable data generation period by the WSN gateways. It is assumed that WSN gateways perform data aggregation and processing of sensor data before forwarding it to remote users. From the mobile network point of view, the WSN gateways are seen as standard mobile users generating traffic.

Several data delivery models, e.g., event driven, query driven or periodic, can be used by the WSN networks internally as well as by the WSN gateways to transfer data across WCDMA radio access network [1]. Our initial analysis includes periodic data delivery method only. This method was selected because we expect that it will result, on average, in the highest traffic load and will have the biggest impact on the radio network.

27.4.1 Control and Automation Domain Applications

Control and automation domain applications provide distributed control and automation in both industrial and residential environments [7]. A typical use case in the industrial domain is monitoring of machines and production equipment, both indoor and in the field. Management of heating and air-conditioning systems, lighting, and security in residential and non-residential buildings is an example of building automation applications [27].

Typically, the machinery monitoring applications use a number of sensors per monitored machine. The frequency of sampling and reporting depends on the machinery complexity and their current status but is generally considered to be a high-frequency process [39]. For the use case of interest, i.e., where the data is to be sent from a WSN gateway to a monitoring center using mobile network as a backbone, we will assume that data packets of average size of 50 bytes are generated by the WSN gateway once a minute.

HVAC (heating, ventilation and air-conditioning control) applications include monitoring and control of industrial and residential premises. A significant number of such applications will use an internal network or fixed Internet to establish communication between the monitored locations and the control center [31]. However, it is expected that mobile networks may be used in some scenarios, especially for remote buildings and buildings without fixed telecom infrastructure. In the commercial domain, we assume that 10 Hz sampling frequency is used. The obtained data is then aggregated and forwarded to control centers every 30 s, using 60-byte long packets [28].

In the residential domain, the sampling is estimated to take place every 30 s [31]. This resulted in an assumption that 60-byte long packets are sent once a minute by the WSN gateway to the control centers. The automatic gas/water/electricity metering applications (AMR) were assumed to send files of up to 2.5 kB twice a day. This is based on the assumption that a reading was done using 15-byte raw data per utility, every 15 min.

27.4.2 Transport Applications

Applications addressing safety of drivers and passengers, traffic monitoring, road toll, and fleet management are typical use cases in the transportation domain [27]. In our analysis, we examined the use cases where mobile terminals (built in the vehicles) in the role of WSN gateways report about the driver's vehicle state, state of the road, and traffic conditions (e.g., monitoring lane occupancy, vehicle count and travel speeds). The measurements are sent every minute by the WSN gateways using 50-byte long data packets [30, 34].

The second considered application in this domain is vehicle tracking application, already being deployed either as a part of corporate processes (security, asset management) or as a car insurance tool. In this application the assumption was that WSN gateways send 50 bytes every 15 s.

27.4.3 Environmental Monitoring for Emergency Services

In the environmental monitoring domain, the flood detection and structural integrity monitoring applications were selected. Environmental processes do not require high-frequency sampling as control and automation processes. For the purpose of our analysis we assume that the sensors will collect 15 bytes of data every 30 s [24]. Gateways will collect and aggregate the data and forward it to the monitoring centers every 3 h. The estimated average file size sent to monitoring centers is 5.4 kB.

27.4.4 Health Monitoring Application Traffic Model

In the health-care domain, the personal health monitoring systems were taken into consideration. These systems use mobile devices to transfer physiological measurements from the sensors attached to a patient's body or located in their immediate vicinity. We assumed that an electrocardiogram (ECG) is monitored using a 3-channel ECG system with 250 Hz sampling rate per channel. Data is transmitted to a monitoring center every 6 h. The amount of data in this case is around 16 Mbytes, which results in 5.4-MByte files every 6 h, with ECG compression rates of 3:1.

27.4.5 Traffic Model Summary

Table 27.1 gives a summary of selected applications, estimated data sizes, and frequencies of data transfer. For all applications interactive traffic class has been adopted and interactive PS radio bearer selected. The frequency of data transfers spans periods of 15 s to 3 h with packet sizes ranging from 50 bytes to few megabytes.

Table 27.1 Selected WSN applications and traffic model

Application	Data size and frequency
Automatic meter reading	2.5 kB every 12 h
Transport	50 B/min or 50 B/15 s
Environmental monitoring	5.5 kB/3 h
HVAC commercial or residential	60 B/30 s or 60 B/min
Health monitoring	5.48 MB/6 h
Machinery monitoring	50 B/min

27.5 Characteristics of the WCDMA Networks

In this section we briefly introduce the main principles and concepts of the mobile networks architecture and protocols to allow easier understanding of the detailed analysis provided in the following sections.

The overall WCDMA mobile networks architecture is given in Fig. 27.1. Mobile terminals (UE — user equipment) interact with radio base stations (NodeB) over the air interface. A number of NodeBs are controlled by a radio network controller (RNC) which on the other hand interacts with the core network elements. When a user wants to set up a call or access Internet, his UE informs the NodeB it's attached to and the corresponding RNC about the required service. Based on this information appropriate resources gets allocated along the path and the service becomes available. Depending on the service needs, different types of radio bearers get allocated. The 3rd Generation Partnership Project (3GPP)[1] R99 release specifications have defined a number of bearers that can be broadly categorized as CS (circuit switched) and PS (packet switched). The PS channels are primarily used for data transfer and are characterized by the throughput they provide. The highest throughput available is provided by the PS384 bearer which provides throughput, as the name implies, of 384 kbit/s. 3GPP Release 5 and Release 6 have introduced high-speed packet access which provides up to 14 Mbit/s in the downlink (HSDPA — High Speed Downlink Packet Access) and up to 5.8 Mbit/s in the uplink (EUL — Enhanced Uplink). Coupled with improvements in the radio access network for continuous packet connectivity, HSPA+ will allow uplink speeds of 11 Mbit/s and downlink speeds of 42 Mbit/s within the 3GPP Release 8 time frame.

[1] www.3gpp.org

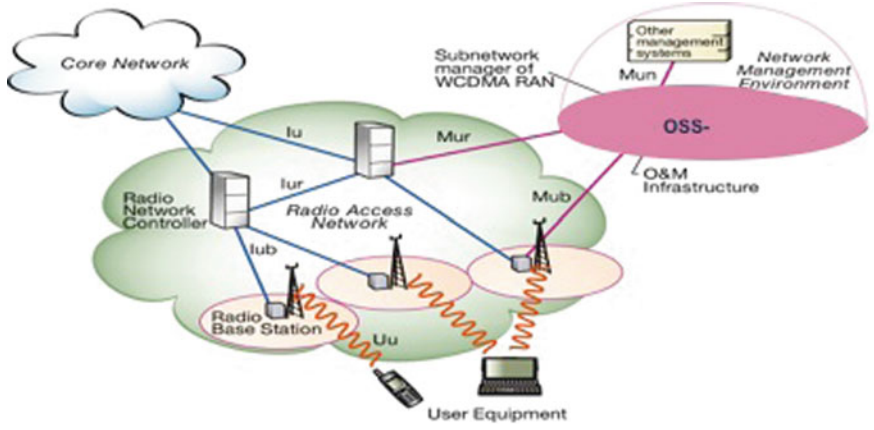


Fig. 27.1 The WCDMA mobile networks architecture

Two types of channels are available to WCDMA users: common and dedicated. Common channels are shared by all users in a radio cell, have a low setup delay, and are used to carry control information to and from the UEs and to transfer short user packets. If a user needs to transfer more data and with a specified quality of service, a dedicated channel (DCH) is set up and allocated to the user. DCH setup is performed based on the signaling exchanged between the user's UE, NodeB the UE is attached to, and the handling RNC.

3GPP defines four traffic classes in WCDMA networks: conversational, streaming, interactive, and background [1]. Each of these classes is mapped onto a RAB, which provides a specific level of quality of service defined by a number of parameters such as transfer delay, delivery order, guaranteed bit rate. Each application is mapped to one of the traffic classes based on its requirements. For example, the voice application is mapped to the conversational class, while web browsing and e-mail download belong to more delay-tolerant classes — i.e., interactive and background.

In [1], examples of applications and the corresponding end-user performance expectations are outlined, using the classification based on the four traffic classes. Telemetry applications have been identified as being suitable to fall into the following traffic classes:

Conversational (real-time): for two-way control telemetry (e.g., controlling an important industrial process)
Streaming (non-real time): for monitoring services

The WSN applications considered here, bar the ones generating frequent transfers of small amount of data, are mapped onto the interactive class. Selected applications in the domains of automatic meter reading and health and environmental monitoring typically do not have strict delay requirements (unless real-time monitoring is required) and therefore are mapped onto interactive class.

Interactive class can be also used for WSN applications that are delay sensitive (e.g., applications reporting on industrial process control and monitoring). These applications typically send periodic updates containing a small amount of data. If a dedicated channel (DCH) is required to be setup to transport such small amount of data, a delay due to the time required to set up such a channel is introduced which might be prohibiting. In addition, using DCH to transfer small amounts of data also results in poor utilization of radio resources (user data to signaling ratio is very small). Hence, when there is no existing DCH channel carrying interactive traffic, these applications may use common channels — forward access channel (FACH) and random access channel (RACH). Of particular interest to M2M and WSN applications is performance of the uplink common channel (RACH), particularly in relation to delay and interference as a function of the number of the users in a cell.

For the purpose of the network dimensioning evaluation we have considered that UEs are in the so-called CELL_DCH state, i.e., dedicated channels are already set up and are ready to be used by WSN applications. The aggregated amount of data generated by the above-mentioned WSN applications during BH is calculated and mapped onto different PS interactive RABs.

For the applications that generate frequent transfers of small amount of data we used common channels for data transfer. This type of traffic is expected to be common in future context-aware networks where all devices interacting with the physical world (sensors, actuators) are registering their capabilities in a registry or directory so that the users of the system can search and find devices capable of answering their queries. For example, the FP7 SENSEI architecture envisages existence of a resource directory as the core functionality [13]. The resource directory stores descriptions of all sensors, actuators, and other devices in the network. It is the responsibility of these devices to keep the description up to date and to regularly inform the resource directory about their availability. This approach will result in a huge number of periodic messages extending the lifetime of records in the directory or updating description due to changes in the environment (in the case of a mobile sensor this could mean updating location).

As the amount of data per transfer is small it is important to keep signaling overhead as low as possible to optimize the transfer. Since the common channels are always on and can be used without additional signaling required to establish dedicated channels, this type of channels seem to be more suitable for short bursts of data. However, the capacity of common channels is limited, and therefore it is necessary to evaluate the overall performance of the system when WSN traffic is sent over these channels. As the data is primarily originating from the edge devices and is transferred to the core network entities, of particular interest is performance of the uplink common channel (RACH). The most important parameters considered in this case are the delay in data transfer and interference as a function of the number of users in the cell (as RACH is shared by all users in the cell). Therefore, only the uplink channel was modeled.

27.6 Network Dimensioning Methodology

Radio network dimensioning is a complex task that depends on many parameters. In this process, the main input parameters that include projected number of network users, applications that will be used, required quality of service for each application, and the size of the area and the type of the environment where the network will be deployed are analyzed. The ultimate goal of the radio access network dimensioning phase is an indication of the size of the radio access network, i.e., number of sites (number of Node B) and the site configuration. This phase is normally followed by the planning phase. The main input parameters used in our study include

- Coverage area and the number of subscribers: 300 km^2 with 455,000 mobile subscribers (an average EU city);
- Average user traffic profile: Expected Circuit-Switched (CS), High Speed Packet Access (HSPA), and Release 99 Packet Switched (PS) traffic has been defined in the form of average subscriber profile during busy hour (BH);
- Site configuration: Node B configuration is a standard solution (20 W) with 3×1 (three sectors and one carrier per sector);
- Number of codes: 10 HSPA codes are used in dense urban, urban and suburban environment and 5 in rural environment.

Six different scenarios were used:

Scenario 1: Standard traffic mix only (no WSN traffic). This scenario was used as a benchmark in the analysis

Scenario 2: WSN-generated traffic only. WSN gateways act as mobile subscribers (same number as in scenario 1), assuming equal WSN application weighing (all WSN applications weigh 1). This scenario was split into two sub-scenarios:

Scenario 2A — WSN traffic only, mapped onto PS64
Scenario 2B — WSN traffic only, mapped onto EUL/HSDPA

Scenario 3: Combined scenarios 1 and 2 traffic profiles, with WSN application penetration 10 and 50%, respectively. WSN application domains are assumed to be equally distributed, i.e., carry the same weight across the application domains and all WSN gateways use all selected WSN applications equally

3A1 — 10% WSN penetration, using PS64 bearer
3A2 — 50% WSN penetration, using PS64 bearer
3B — 50% WSN penetration using EUL/HSDPA RABs

Scenario 4: Combined standard and WSN traffic with non-equal distribution of WSN applications. WSN traffic was mapped to PS64 RAB and EUL/HSDPA RABs in sub-scenarios 4A and 4B, respectively. The overall WSN penetration of 50% with the following weights based on the expected penetration rate of different application domains [2] was used:

- Control & Automation: 0.744 (Machine Monitoring 0.35, Advanced Metering 0.18, HVAC Residential 0.18, HVAC Commercial 0.29)
- Transport: 0.083 (equally split among two use cases)
- Emergency Monitoring: 0.1
- Health: 0.079

Scenario 5: Combined standard and WSN traffic with equal application mix and 100% WSN penetration, using EUL/HSDPA bearers

Scenario 6: Combined standard and double WSN traffic with equal application mix and 200% penetration, using EUL/HSDPA bearers

In scenarios 3, 4, 5, and 6, the average traffic profile was normalized across 455,000 subscribers, taking into account standard and WSN application mix as well as the penetration levels of WSN applications.

The performance of a RACH channel (common uplink channel) is simulated in an Ericsson simulation tool. The following simulation parameters were used:

- Simulation time: 400 s
- Sources of traffic: five per M2M client generating traffic every 15, 30, and 60 s
- Packet sizes: 50, 100, 150, and 200 bytes
- Number of users per cell: 1, 40, 80, 160, 320, and 640.

The following parameters were monitored:

1. One-way delay (uplink): this is a delay the packet experiences from the moment it is sent by the client until its reception by the server.
2. RACH access delay: this is the time since the first RACH preamble is sent until the acknowledgment is received by the mobile user, i.e., until data transfer can start.
3. Attempts by the UE: total preamble attempts before the data is actually transmitted.
4. Uplink interference: level of interference on the uplink.

These parameters were selected as they give a good insight into the performance of the RACH channel both from user perspective (delay) and from the system's point of view (uplink interference) [1]. The combination of delays in accessing RACH channel and the data transfer delay can have a detrimental impact on the application while the uplink interference has impact on the overall capacity of the mobile network.

27.7 Results

The main results of a radio access network dimensioning exercise are the number of (radio) sites required, site configuration, traffic per site/cell, and capacity per site/cell. Figure 27.2 depicts the total number of required sites for all scenarios.

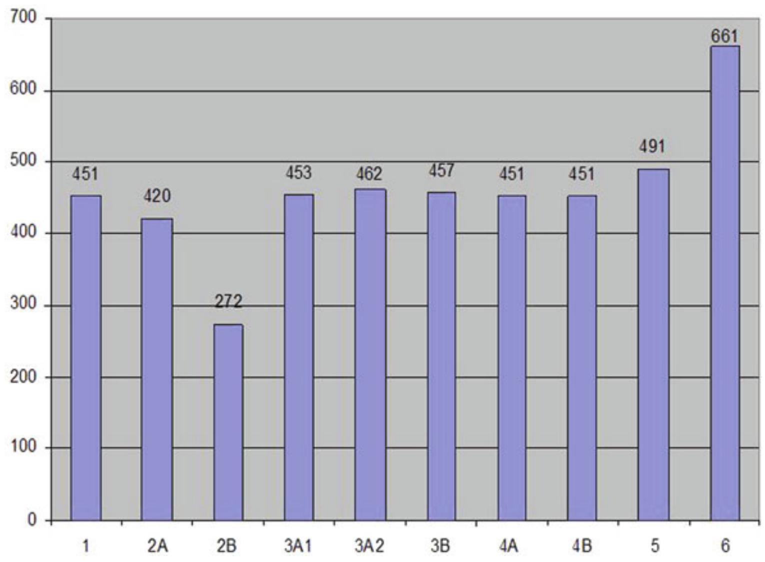


Fig. 27.2 Total number of sites

In relation to the results of scenario 1, the following can be noted:

- Required number of sites to support 455,000 users with standard traffic mix is 451. This value is used as a benchmark for analysis of scenarios involving WSN traffic.
- WCDMA RAN dimensioning process envisages separate calculation of the required network resources for uplink and downlink; the final allocation of resources is done according to the higher requirements of the two. In this scenario, the size of the network is determined by the downlink (DL) requirements in terms of the number of sites as the standard traffic mix is downlink heavy.
- The size of the network is also impacted by a near-100% utilization of the code tree in most environments, considering that each WCDMA cell has a limited number of codes available.

In contrast to scenario 1, the size of the network in scenario 2 is determined by the uplink requirements, which was expected considering the nature of WSN traffic (uplink mainly). The capacity per site and the number of sites quite differ between scenarios 2A and 2B, despite the fact that the same traffic mix is used. This is due to the fact that WSN traffic in scenario 2A uses PS64 RAB, and in scenario 2B it uses EUL/HSDPA RABs. Usage of PS64 RAB results in high traffic intensity (Erlang/subscriber — the amount of traffic generated by a subscriber over a period of time, usually 1 h), which reduces capacity per site and hence increases the number of sites. In scenarios 3A2 and 3B, the number of sites is determined by the requirements for the DL traffic. This means that the increase in the number of sites on the uplink due to WSN traffic, with a given WSN penetration, still does not outweigh the number of sites required to support the downlink traffic.

The traffic mix in scenarios 4A and 4B, which take into account different application penetration levels, is determined on the uplink mainly by the WSN-generated traffic and on the downlink by the standard mix. The number of sites to support the downlink traffic again outweighs the number of sites on the uplink; therefore, their results require networks of the same size as in scenario 1.

Scenarios 5 and 6 see an increase in the required number of sites compared to scenario 1 due to the increase of traffic per subscriber. However, it should be noted that out of all scenarios that feature standard traffic profile (scenarios 3, 4, 5, and 6), only in scenario 6, where WSN traffic is doubled, the resulting number of sites is determined by the requirements on the uplink. Figure 27.3 shows relative increase in the traffic intensity per subscriber during BH for scenarios 3, 4, 5, and 6 when compared to scenario 1. Relative increase in the traffic intensity (Erlang) per subscriber is particularly noticeable in scenario 3A2 due to the use of PS64 bearer and is even higher than in scenario 6 in which the WSN traffic is doubled and EUL/HSDPA RABs are used.

Relative change in the total hardware processing capacity required across all sites on the uplink and downlink on Node Bs to handle R'99/HSPA traffic is given in Fig. 27.4. It should be noted that the required processing resources at Node Bs depend on the characteristics of the actual hardware components and vary between the vendors. Hence, only relative increases are depicted by this figure.

As we can see from Figs. 27.2, 27.3, and 27.4, significant increase in scenario 3A2 in Erlang per subscriber (as a consequence of additional WSN-related traffic and the use of PS64 bearer to carry it) has resulted in a correspondingly significant increase of the total required processing capacity both on the uplink and on the

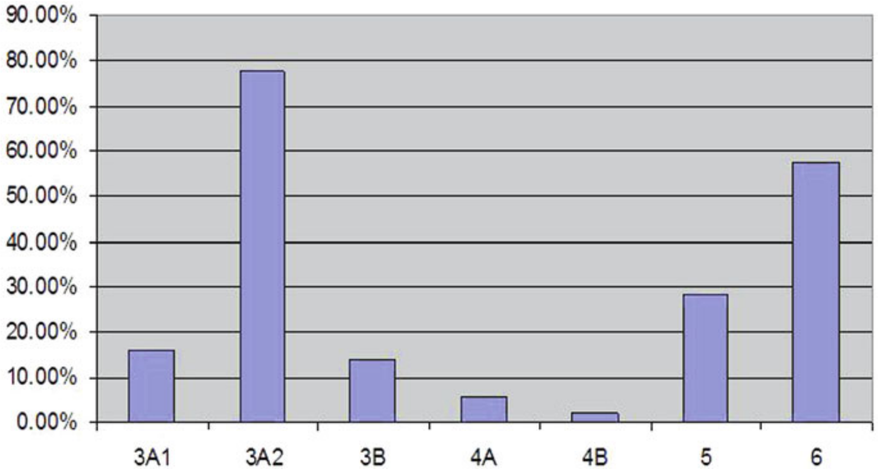


Fig. 27.3 Relative increase in traffic demand (Erlang) per average user during BH — scenarios 3, 4, 5, and 6

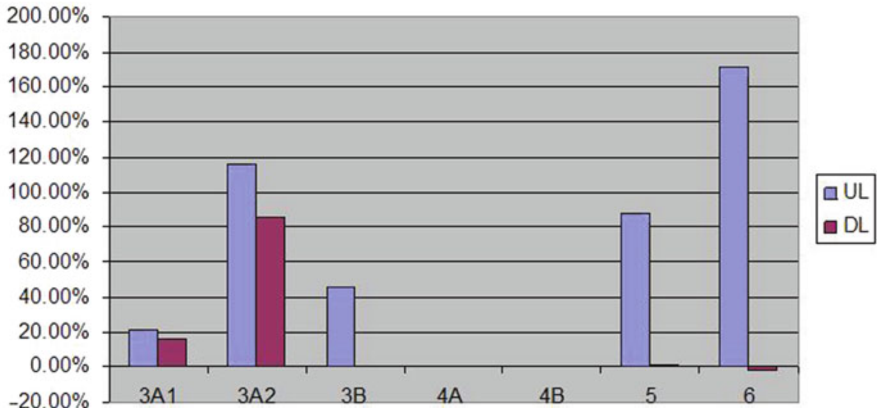


Fig. 27.4 Relative change of the total required R'99/EUL processing resources compared to scenario 1

downlink. Scenario 3B features the same WSN traffic profile, however with a much lower Erlang/subscriber increase (14%) compared to scenario 3A2. This comes as a result of using high-speed RABs — EUL/HSDPA. An increase of traffic intensity is observed mainly on the uplink due to the nature of WSN traffic, hence the increase in the use of total uplink R'99/HSPA resources.

In scenarios 4A and 4B a very small increase in Erlang/subscriber (5.9 and 2%, respectively) is introduced, which explains no requirement for additional processing capacity.

Scenarios 5 and 6 both feature high-speed bearers — EUL/HSDPA to carry the additional WSN traffic assumed to be equally distributed among the WSN applications. With 100 and 200% WSN penetration for scenarios 5 and 6, respectively, the normalized traffic profile on the uplink is entirely determined by the WSN traffic, resulting in significant increase in number of sites and total processing R'99/EUL resources on the uplink. It should be noted, however, that only in scenario 6, the number of sites is determined by the uplink requirements (except for suburban environment).

All HSPA resources that are in place to support standard traffic mix (i.e., codes, EUL license, HS-capable boards) are reused in scenarios 3B, 4B, 5, and 6.

27.7.1 Common Channels Analysis

One-way data transfer delay as a function of the number of users in a cell using RACH for different packet sizes (50, 100, 150, and 200 bytes) is presented in Fig. 27.5. It can be noted that for smaller packet sizes, the delay increases linearly and relatively slow (from 90 up to 110 ms for 50-byte packets) with the increase in the numbers of users in a radio cell (from 1 to 640), while the curves are getting

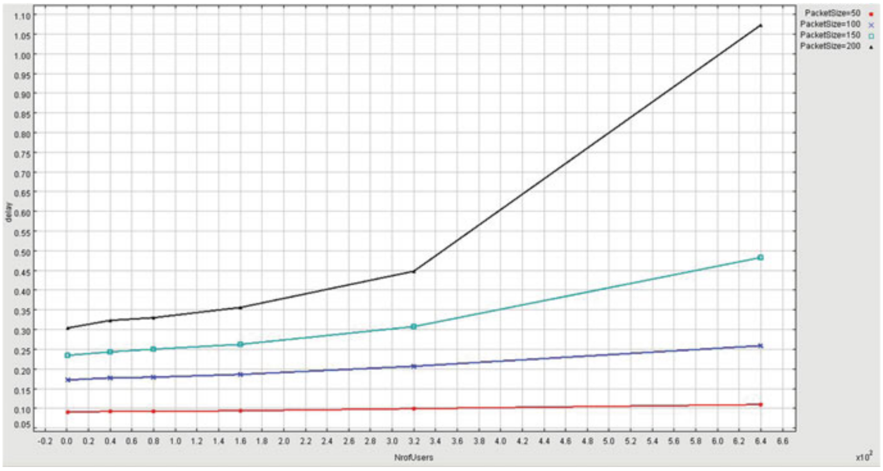


Fig. 27.5 One-way delay for different packet sizes

steeper as the packet sizes get bigger. The delay reaches up to 1.1 s when packet size of 200 bytes is used with 640 users in the cell.

Delay of 100 ms is a relatively short one and will be acceptable for a range of applications. However, delays of more than 1 s, particularly for applications requiring return control information, will most likely be exceeding the delay budget as these applications would typically require a sub-second end-to-end delay.

Figure 27.6 presents RACH channel access delay. RACH channel delay is the time from the moment the first RACH preamble is sent until the acknowledgment is received by the mobile user. In tested scenarios, this delay ranges from 2 to 16 ms,

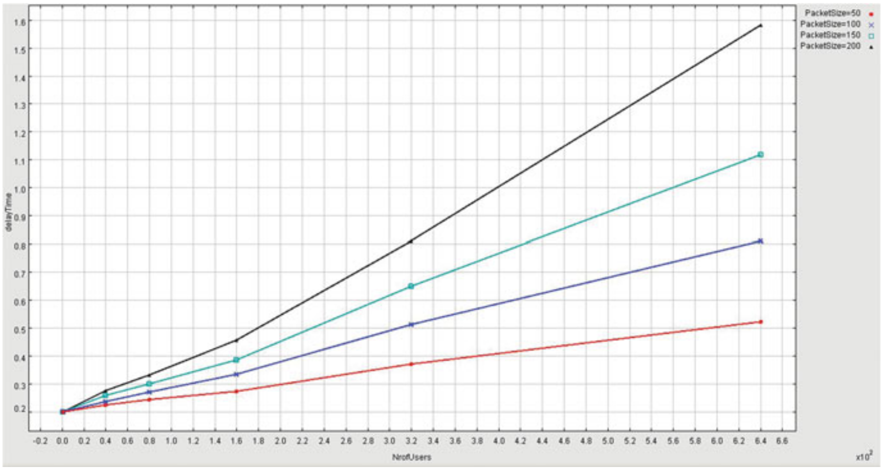


Fig. 27.6 RACH channel access delay

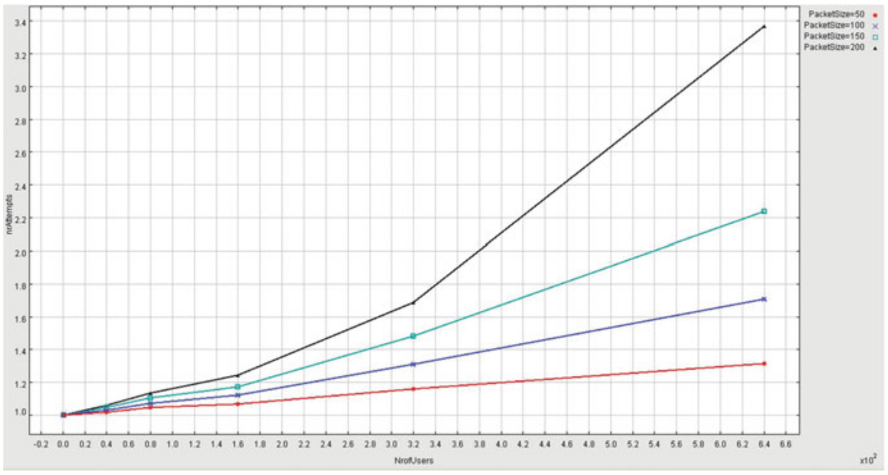


Fig. 27.7 RACH channel — attempts by UE

depending on the number of users and packet sizes. This delay is relatively small in comparison with the one-way delay presented in Fig. 27.5. Therefore, it does not present a problem from the M2M application point of view and can be disregarded.

The mean number of total preamble attempts before data is transmitted increases with the increase in the number of users and packet sizes and ranges on average 1.4–3.4 (Fig. 27.7). It should be noted that the maximum number of preamble attempts by user equipment (UE) per cycle is normally set to 8 (default), while maximum number of cycles is normally set to 4, before RACH procedure is aborted. It can be concluded that this factor does not impact execution of data transfers apart from adding a small delay which can be disregarded in comparison to the one-way delay presented in Fig. 27.5.

Figure 27.8 shows interference on the uplink as the function of the number of users in a radio cell and packet sizes. While previous graphs are showing various data transfer delays introduced by the network and giving information on suitability of RACH channel for M2M traffic from the application point of view, the interference graph is showing impact of M2M traffic on the overall network performance as interference is one of the main limiting factors for the uplink.

Typical thermal noise interference for WCDMA is -105 dB m including the receiver noise. With the traffic growth and increased network load interference is increasing, which is defined as noise rise. Due to the increased interference the cell breathing phenomenon occurs in WCDMA systems, which is visible as radio cell coverage shrinking. Usually, mobile systems are designed to cope with the noise rise of up to 5 dB. As it can be seen in Fig. 27.8, for packet size of 50 bytes interference on the uplink is always acceptable, i.e., less than -100 dB m. However, for the larger packet sizes (i.e., 100, 150, and 200 bytes), the maximum number of simultaneous users is falling to 320, 220, and 170 users, respectively. This is one of the most

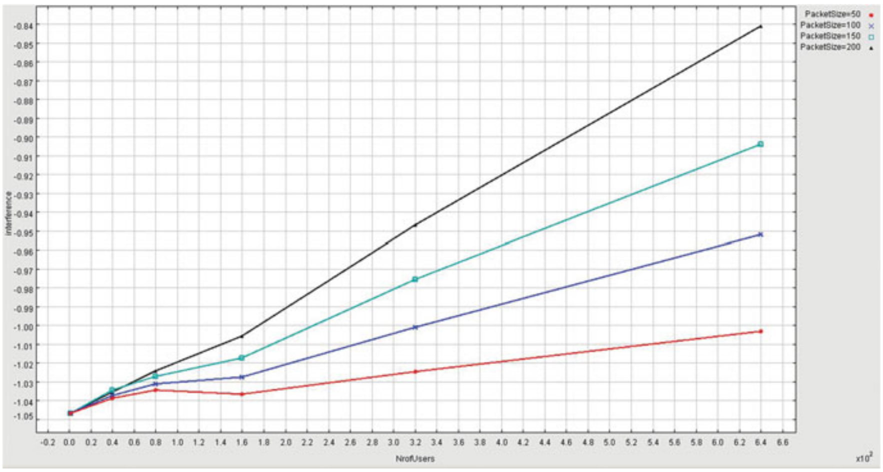


Fig. 27.8 Interference on uplink

limiting factors for the use of RACH channels for M2M traffic as it significantly reduces the number of simultaneous users.

In addition to the impact on radio cell coverage, the growing interference will degrade uplink performance, especially for the EUL, where scheduler allocates resources to EUL users taking into account the uplink interference in the cell. Due to the increased interference, the EUL users will experience degradation of HSUPA performance in terms of achieved throughput and delays. Therefore, usage of RACH channels has to be controlled and its use is limited so that it does not affect other users.

27.8 Conclusions

The study described is one of the first attempts to quantify the potential impact of the so-called *Internet of Things* on mobile access networks. The study used projections of traffic profiles and standard reference values for system parameters. Assumptions in relation to the WSN traffic were based on a number of studies and descriptions of prototype applications, as well as live commercial systems. The analysis encompassed a range of M2M applications, with data size that range from a few kilobytes to more than 1 MByte during BH. All selected WSN/M2M applications were mapped to interactive RABs. The capacity analysis showed that the number of radio sites in a network remains mainly determined by the downlink traffic in scenarios featuring standard traffic mix, except in scenario 6 where WSN traffic was doubled. The observed increase of the required R'99/HSPA resources due to the WSN/M2M traffic is dependent on the bearer use (PS64 or high-speed bearers). The use of EUL/HSDPA bearers is recommended for data-heavy applications. RACH performance, particularly in relation to delay and interference, has to be analyzed separately, as well as the delay when dedicated/HS channels are used. Finally, it should be noted that the number of concurrent users on the radio network controller (RNC) represents a potential bottleneck in the above setup, as all UEs that are also WSN gateways will be in connected mode (in Cell-FACH state when using common channels or in Cell-DCH state when using dedicated channels), i.e., active from the RNC point of view. This aspect will be further explored in the future. Further refinement of the selected WSN traffic models, the actual penetration levels of WSN applications as well as inclusion of new applications like social networking into the analysis are some of the activities planned in the future.

Simulations show one-way packet delay ranging from 110 ms for 50-byte data to 1.1 s for 200-byte data for 640 users per cell when using RACH. The delay budget has to be drawn for every particular M2M application, but in general delays of more than 1 s can be a limiting factor for a range of applications. Intensive usage of RACH for M2M traffic may affect network performance in general and other users, particularly EUL users, due to the increased interference. Interference increases with the number of simultaneous RACH users and data packet sizes which limits the number of M2M users that can simultaneously use RACH. This has to be taken into account when bearer mapping for M2M users is selected.

References

1. 3rd Generation Partnership Project (3GPP). Services and service capabilities. Technical Standard TS 22.105, R99, 3rd Generation Partnership Project (3GPP) Mobile Competence Centre, Sophia Antipolis, France, 1999.
2. K. Aberer, M. Hauswirth, and A. Salehi. Infrastructure for data processing in large-scale interconnected sensor networks. In: *Mobile Data Management, 2007 International Conference*, pages 198–205, Mannheim, Germany, May 2007.
3. K. Aberer, M. Hauswirth, and A. Salehi. A middleware for fast and flexible sensor network deployment. In: *VLDB '06: Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 1199–1202, Seoul, Korea, 2006.
4. D. Alberts, J. Garstka, R. Hayes, and D. Signori. *Understanding Information Age Warfare*, 2nd edition. CCRP Publication Services, Washington, USA, 2001.
5. D. Alberts, J. Garstka, and F. Stein. *Network Centric Warfare: Developing and Leveraging Information Superiority*, 2nd edition. CCRP Publication Services, Washington, USA, 1999.
6. Apple Inc. Nike + ipod: Meet your new personal trainer. online.
7. S. Baydere, E. Cayirci, I. Hacioglu, O. Ergin, A. Olero, I. Maza, A. Viguria, and P. Bonnety. Applications and application scenarios. Project Deliverable IST-004400, Embedded WiSeNts Consortium, IPVS, Distributed Systems Group, Universitat Stuttgart, Universitatsstr. 38, D-70569 Stuttgart, Germany, February 2006.
8. D. E. Bell. Looking back at the bell-la padula model. In: *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC 2005)*, Tucson, AZ, USA, 2005.
9. J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In: *Proceedings of the World Sensor Web Workshop, ACM Sensys 2006*, Boulder, CO, USA, 2006.
10. L. Compagna, V. Lotz, and K. Wrona. Towards adaptive security for ubiquitous computing systems: Mosquito and serenity. In Muhlhauser M., Gurevych I., editors. *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*. Idea Group Publishing, Hershey, USA, 2008.

11. T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel. A survey of lightweight-cryptography implementations. *IEEE Design and Test of Computers*, 24(6):522–533, 2007.
12. Ericsson. Vision 2020—50 billion connected devices. online.
13. A. Gluhak, M. Bauer, F. Montagut, V. Stirbu, M. Johansson, J. B. Vercher, and M. Presser. Towards an architecture for a real world internet. In: *Proceedings of the Autonomous and Spontaneous Networks Symposium*, Telecom ParisTech, Paris, 2008.
14. J. Goldman, K. Shilton, J. Burke, D. Estrin, M. Hansen, N. Ramanathan, S. Reddy, V. Samanta, M. Srivastava, and R. West. Participatory sensing: A citizen-powered approach to illuminating the patterns that shape our world. Technical report, UCLA's Center for Embedded Networked Sensing (CENS), Los Angeles, CA, USA, May 2009.
15. W. I. Grosky, A. Kansal, S. Nath, J. Liu, and F. Zhao. Senseweb: An infrastructure for shared sensing. *Multimedia, IEEE*, 14(4):8–13, October–December 2007.
16. M. Hefeeda and M. Bagheri. Wireless sensor networks for early detection of forest fires. In: *Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems*, pages 1–6, Los Alamitos, CA, USA, 2007.
17. C. Huddlestone-Holmes, G. Gigan, G. Woods, A. Ruxton, I. Atkinson, and S. Kininmonth. Infrastructure for a sensor network on davies reef, great barrier reef. In: *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP)*, pages 675–679, Melbourne, Australia, 2007.
18. R. S. H. Istepanian, S. Laxminarayan, and C. S. Pattichis, editors. *Mhealth: Emerging mobile health systems*. Topics in Biomedical Engineering Series Topics in Biomedical Engineering International Book Series. Springer, New York, USA, 2006.
19. M. Kim, Y. J. Lee, and J. Ryou. What are possible security threats in ubiquitous sensor network environment? In: *Managing Next Generation Networks and Services—Proceedings of the APNOMS 2007*, volume 4773 of *LNCS*, Sapporo, Japan, pages 437–446, 2007.
20. M. Klopfer and I. Simonis, editors. *SANY: An open service architecture for sensor networks*. SANY Consortium, 2009.
21. S. Krco. Health care sensor networks—architecture and protocols. *Journal on Ad Hoc and Sensor Wireless Networks*, 1, January 2005.
22. S. Krco and V. Delic. Personal wireless sensor network for mobile health care monitoring. In: *Proceedings of the 6th International IEEE Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Service (TELSIKS)*, Serbia and Motenegro, October 2003.
23. S. Krco, M. Johansson, and V. Tsiatsis. A commonsense approach to realworld global sensing. In: *Proceedings of the SenseID: Workshop on Convergence of RFID and Wireless Sensor Networks and their Applications, ACMSenSys 2007*, Sydney, Australia, 2007.
24. M. Kuorilehto, M. Hännikäinen, and T. D. Hämäläinen. A survey of application distribution in wireless sensor networks. *EURASIP Journal of Wireless Communication Network*, 2005(5):774–788, 2005.
25. M. A. Laguna, J. Finat, and J. A. Gonzalez. Mobile health monitoring and smart sensors: A product line approach. In: *Proceedings of the Euro American Conference on Telematics and Information Systems: New Opportunities to Increase Digital Citizenship*, Prague, Czech Republic, 2009.
26. D. Liu, P. Ning. *Security for Wireless Sensor Networks*. Springer, Heidelberg, 2007.
27. P. J. Marron and D. Minder. Embedded WiSeNts Consortium. Embedded wisents research roadmap. Project Deliverable IST-004400 D3.3, Embedded WiSeNts Consortium, IPVS, Distributed Systems Group, Universitat Stuttgart, Universitatsstr. 38, D-70569 Stuttgart, Germany, November 2006.
28. T. Melodia, D. Pompili, and I. F. Akyildiz. A communication architecture for mobile wireless sensor and actor networks. In: *Proceedings of the 3rd Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, Reston, VA, USA, 2006.
29. W. M. Merrill, L. Girod, B. Schiffer, D. McIntire, G. Rava, K. Sohrabi, F. Newberg, J. Elson, and W. Kaiser. Dynamic networking and smart sensing enable next-generation landmines. *IEEE Pervasive Computing*, 3:84–90, 2004.

30. S. Murthy. Data communication with remote sensors using reflex narrowband pcs technology. Final Project Report ITS-IDEA Project 74, Transportation Research Board, 500 Fifth Street, NW, Washington, DC, USA, 2001.
31. A. Pinto, L. P. Carloni, and A. L. Sangiovanni-Vincentelli. A communication synthesis infrastructure for heterogeneous networked control systems and its application to building automation and control. In: *Proceedings of the 7th ACM and IEEE International Conference on Embedded Software (EMSOFT)*, Salzburg, Austria, 2007.
32. G. Roladerr. Self healing (antitank) minefield (shm) saic team program. In: *Proceedings of the 2002 Mines, Demolition and Non-Lethal Conference & Exhibition*, Ft. Belvoir, VA, USA, 2002.
33. K. Schleidt and D. Havlik. Sensorweb-based prototype for air quality reporting systems. *ERCIM News*, (79):48–49, October 2009.
34. P. Shankar. Designing an inter-vehicular network stack for car-to-car communication. In: *Proceedings of the Rutgers-Helsinki Ph.D. Student Workshop on Spontaneous Networking*, Piscataway, NJ, USA, 2006.
35. J. Shneidman, P. Pietzuch, J. Ledlie, M. Roussopoulos, M. Seltzer, and M. Welsh. Hourglass: An infrastructure for connecting sensor networks and applications. Technical Report Technical Report TR-21-04, Harvard University, Cambridge, Massachusetts, 2004.
36. A. Sorniotti, L. Gomez, K. Wrona, and L. Odorico. Secure and trusted in-network data processing in wireless sensor networks: A survey. *International Journal of Information Assurance and Security*, 2(4):189–199, 2007.
37. M. Srivastava, M. Hansen, J. Burke, A. Parker, S. Reddy, G. Saurabh, M. Allman, V. Paxson, and D. Estrin. Wireless urban sensing systems. Technical Report Technical Report 65, UCLA's Center for Embedded Networked Sensing Systems (CENS), Los Angeles, CA, USA, April 2006.
38. A. Nayak and I. Stojmenovic. *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*. Wiley, Hoboken, NJ, USA, 2010.
39. V. Sundararajan, A. Redfern, M. Schneider, P. Wright, and J. Evans. Wireless sensor networks for machinery monitoring. In: *Proceedings of the 2005 ASME International Mechanical Engineering Congress and Exposition*, Orlando, FL, USA, 2005.
40. O. Tekdas, J. H. Lim, A. Terzis, and V. Isler. Using mobile robots to harvest data from sensor fields. *IEEE Wireless Communications*, 16(1):22–28, 2009.
41. T. Datagate Inc. Interactive link data diode device. Common Criteria Security Target Doc. No. 9162P01000001, Issue No. 5.1, National Information Assurance Partnership (NIAP), Vrginia, VA, USA, 2005.
42. G. Tselentis, J. Domingue, A. Galis, A. Gavras, D. Hausheer, S. Krco, V. Lotz, and T. Zahariadis. *Towards the Future Internet—A European Research Perspective*. IOS Press, Amsterdam, The Netherlands, 2009.
43. U. Varshney. Pervasive healthcare and wireless health monitoring. *Journal of Mobile Networks and Applications*, 12(2–3), June 2007.
44. Vital Wave Consulting. mhealth for development: The opportunity of mobile technology for healthcare in the developing world. Technical report, United Nations Foundation and Vodafone Foundation Technology Partnership, New York, NY, USA, 2009.
45. P. Volgyesi, G. Balogh, A. Nadas, C. B. Nash, and A. Ledeczi. Shooter localization and weapon classification with soldier-wearable networked sensors. In: *Proceedings of the 5th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Juan, Puerto Rico, 2007.
46. M. Weiser. The computer for the twenty-first century. *Scientific American*, 265(3):94–104, 1991.
47. M. Winkler, K.-D. Tuchs, K. Hughes, and G. Barclay. Theoretical and practical aspects of military wireless sensor networks. In: *Proceedings of the Military Communications and Information Systems Conference*, Bonn, Germany, 2007.