Studies in Systems, Decision and Control 163

Habib M. Ammari *Editor*

Mission-Oriented Sensor Networks and Systems: Art and Science

Volume 1: Foundations

Springer

Studies in Systems, Decision and Control

Volume 163

Series Editor

Janusz Kacprzyk, Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland

The series "Studies in Systems, Decision and Control" (SSDC) covers both new developments and advances, as well as the state of the art, in the various areas of broadly perceived systems, decision making and control–quickly, up to date and with a high quality. The intent is to cover the theory, applications, and perspectives on the state of the art and future developments relevant to systems, decision making, control, complex processes and related areas, as embedded in the fields of engineering, computer science, physics, economics, social and life sciences, as well as the paradigms and methodologies behind them. The series contains monographs, textbooks, lecture notes and edited volumes in systems, decision making and control spanning the areas of Cyber-Physical Systems, Autonomous Systems, Sensor Networks, Control Systems, Energy Systems, Automotive Systems, Biological Systems, Vehicular Networking and Connected Vehicles, Aerospace Systems, Automation, Manufacturing, Smart Grids, Nonlinear Systems, Power Systems, Robotics, Social Systems, Economic Systems and other. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution and exposure which enable both a wide and rapid dissemination of research output.

** Indexing: The books of this series are submitted to ISI, SCOPUS, DBLP, Ulrichs, MathSciNet, Current Mathematical Publications, Mathematical Reviews, Zentralblatt Math: MetaPress and Springerlink.

More information about this series at http://www.springer.com/series/13304

Habib M. Ammari
Editor

Mission-Oriented Sensor Networks and Systems: Art and Science

Volume 1: Foundations

Springer

Editor
Habib M. Ammari
Wireless Sensor and Mobile Ad-hoc Network
Applied Cryptography Engineering
(WiSeMAN-ACE) Research Lab
Department of Electrical Engineering
and Computer Science
Frank H. Dotterweich College of Engineering
Texas A&M University-Kingsville
Kingsville, TX, USA

ISSN 2198-4182 ISSN 2198-4190 (electronic)
Studies in Systems, Decision and Control
ISBN 978-3-319-91145-8 ISBN 978-3-319-91146-5 (eBook)
https://doi.org/10.1007/978-3-319-91146-5

Library of Congress Control Number: 2018941995

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To Allah, the Most Beneficent, the Most Merciful; and His Prophet, Mohamed, Peace Be Upon Him

To my first teachers: My mother, Mbarka, and my father, Mokhtar.

To my very best friends: My lovely wife Fadhila, and beautiful children, Leena, Muath, Mohamed-Eyed, Lama, and Maitham.

To my dearest brother, Lazhar, and sisters, Naima, Saloua, Monia, Faouzia, Alia, and Najeh.

To all my wonderful nieces, Rahma, Safa, Marwa, Amani, Omayma, and Sabrine; and my nephews, Mohamed, Bilel, Ahmed, and Youssef.

To the profound souls of my grandparents, Abdelkarim and Fatma, and my uncle, Mahfoudh.

To my Dean, Dr. Mohammad S. Alam, Fellow-IEEE, IET, OSA, SPIE, IoP, IS&T, and IAPR, Professor of Electrical Engineering, in the Department of Electrical Engineering and Computer Science, Frank H. Dotterweich College of Engineering, at Texas A&M University-Kingsville, for his outstanding support to me since I joined Texas A&M University-Kingsville in August 2019.

To Dr. Afzel Noore, Associate Dean for Undergraduate Affairs, and Professor of Computer Science in the Department of Electrical Engineering and Computer Science, Frank H. Dotterweich College of Engineering, at Texas A&M University-Kingsville, for his excellent support to me since I joined Texas A&M University-Kingsville in August 2019.

To the profound soul of my Professor, Dr. Hedi Ben Saad, Professor of Mathematics in the Department of Mathematics at the Faculty of Sciences of Tunis, Tunisia (May 25, 1950–June 30, 2016). He taught me mathematics in my second year of Physics and Chemistry major at the Faculty of Sciences of Tunis during the academic year 1987–1988. He was extremely knowledgeable in mathematics, very humble, and so kind. I have never seen anyone so far in his goodness, humility, and acute intelligence in mathematics.

To the profound soul of my Provost, Dr. Stephen Freedman (April 7, 1950–July 2, 2018), Professor of Biology and Senior Vice President for Academic Affairs and Chief Academic Officer at Fordham University, for his outstanding support to me during my stay at Fordham University.

To all of my friends and colleagues in the Department of Electrical Engineering and Computer Science, Frank H. Dotterweich College of Engineering, at Texas A&M University-Kingsville, for their wonderful friendship and outstanding support to me since I joined Texas A&M University-Kingsville in August 2019.

Foreword

Mission-oriented sensor networks have moved from the research domain into practical settings over the past decade. These networks are deployed to fulfill a specific mission, or a number of specific types of missions. As such, the sensors and networks are tailored to meet mission needs in terms of sensing, coverage, lifetime, and security. With different applications come different requirements on network connectivity, capacity, processing, and security. These types of applications often lead to application-specific solutions. However, there has been great progress in the creation of general platforms, protocols, and systems that can meet the needs of many types of mission. This is helpful for making systems more robust and cost-effective. Still, much of the potential for such networks is untapped, as technical problems remain open for many practical applications.

The first volume of this book provides a comprehensive coverage of the major technical challenges in mission-oriented sensor networks written by an outstanding set of experts. The topics covered range from the architecture of sensor nodes to system-level issues such as coverage and mobility, and allocation of sensors to missions. These high-level systems issues can be generalized across many applications. The book also covers important specific technology topics such as localization and tracking, data dissemination, routing, and security and privacy. This is a timely book that will be a valuable reference as mission-oriented sensors networks are deployed for a more wide variety of applications, and the innovations of the past are adapted for multiple uses.

The first several chapters of Volume 1 discuss practical aspects of wireless sensor nodes and networks. They point out that WSNs are application-specific and such system must be designed to support specific applications and be resilient. In particular Chapters "Design Considerations of Mission-Oriented Sensor Node Architectures", "Failure Handling in RPL Implementations: An Experimental Qualitative Study", and "On the Optimization of WSN Deployment for Sensing Physical Phenomena: Applications to Urban Air Pollution Monitoring" cover application-specific node architectures, resilient routing in WSNs, and deployment in urban environments for pollution detection.

In mission-oriented sensor networks, there may be competition for resources. If multiple missions are ongoing at once, resources must be assigned or scheduled in a way to maximize the usefulness of the network and to conserve energy. In addition, such networks must be useable so natural language interfaces are very attractive. Chapters "Energy-Aware Task Allocation in WSNs", "Sensor Assignment to Missions: A Natural Language Knowledge-Based Approach", and "Resource Allocation and Task Scheduling in the Cloud of Sensors" cover these topics.

One important service that must be provided to many sensor network applications is localization. In some cases, it is known where sensors are deployed so the location from which readings occur is also known. But in many others, sensors are deployed on mobile platforms, such as buses, animals, or may be self-moving. In these cases, data must be labeled with the location from where it was generated. In addition to localizing themselves, it is important for sensors to localize the object or phenomena they are sensing. Chapters "Target Detection, Localization, and Tracking in Wireless Sensor Networks", "Regularization-Based Location Fingerprinting", "Sense-Through-Foliage Target Detection Based on UWB Radar Sensor Networks", and "Mobile Target Tracking with Multiple Objectives in Wireless Sensor Networks" cover localization of sensors and targets, and target tracking in many environments.

A critical aspect of sensor networks is collecting the information gathered by sensors at a point where it can be processed or used. This requires routing information from the sensors to a sink. To save on resources, or add value to individually gathered data, data fusion is a popular technique. This requires specialized routing. In general, routing and topology control will have large impacts on the performance of data gathering in sensor networks as well as energy efficiency. The problem of data diffusion and gathering becomes even more complex in wireless sensor networks in which nodes can move. Both topology control and routing are affected by mobility. In some cases, delay-tolerant data collection may be required. Chapters "Data Dissemination and Remote Control in Wireless Sensor Networks", "A Data Fusion Algorithm for Multiple Applications in Wireless Sensor Networks", "Underwater Networks for Ocean Monitoring: A New Challenge for Topology Control and Opportunistic Routing", "Geometric Routing Without Coordinates but Measurements", and "Delay-Tolerant Mobile Sensor Networks: Routing Challenges and Solutions" cover these very important topics.

Mission-oriented sensor networks are expected to be used in defense, law enforcement, healthcare, and corporate applications such as advanced manufacturing. Therefore, security is of paramount importance. Everything from the location of a sensor to the data it is gathering and transmitting must be protected from eavesdropping and tampering. Likewise, sensor networks must be protected against data being injected into their domain. Many standard security solutions, such as those related to cryptography, are too complex to be implemented on simple sensor nodes. These challenges are even greater in wireless sensor networks. Chapters "Location Privacy in Wireless Sensor Networks", "Implementation of Secure Communications for Tactical Wireless Sensor Networks", "Data-Driven Detection

of Sensor-Hijacking Attacks on Electrocardiogram Sensors", and "Cryptography in WSNs" cover these topics.

This volume presents both an overview of the state of the art of these important topics and research approaches for solving important open problems. They cover specific applications for sensor networks, such as monitoring air pollution outdoors, or EKGs in a body, that illustrate the wide range of challenges. These chapters are very timely given that sensor networks are on the cusp of widespread deployment and use.

March 18, 2019

Thomas F. La Porta
Director, School of Electrical Engineering

Evan Pugh
Professor, Penn State University

William E. Leonhard
Professor, Penn State University

Contents

Part VII Privacy and Security

About the Editor

Habib M. Ammari is an Associate Professor and the Founding Director of Wireless Sensor and Mobile Ad-hoc Network Applied Cryptography Engineering (WiSeMAN-ACE) Research Lab, in the Department of Electrical Engineering and Computer Science, Frank H. Dotterweich College of Engineering, at Texas A&M University-Kingsville (TAMUK) since August 2019. He received his tenure in May 2014 in the Department of Computer and Information Science, College of Engineering and Computer Science, at the University of Michigan-Dearborn, where he served on the Distinguished Research Award Committee in 2015. Also, he received tenure at the Higher School of Communications in Tunis, Tunisia (Sup'Com Tunis) in 1998. Recently, he received the 2018 Albert Nelson Marquis Lifetime Achievement Award. He was selected as instructor at Stanford University in the Stanford Summer College Academy 2016 program, where he taught "Discrete Mathematical Structures: Foundational Concepts in Computer Science, Engineering, and Mathematics". He obtained his second Ph.D. degree in Computer Science and Engineering from the University of Texas at Arlington, in May 2008, and his first Ph.D. in Computer Science from the Faculty of Sciences of Tunis, in December 1996. His main research interests lay in the area of wireless sensor and mobile ad hoc networks, including connected *k*-coverage, geographic forwarding, physical and information security, applied cryptography, and computational geometry in wireless sensor networks. He has a strong publication record in top-quality journals, such as ACM TOSN, ACM TAAS, IEEE TPDS,

IEEE TC, Elsevier Ad Hoc Networks, Elsevier COMNET, Elsevier PMC, Elsevier JPDC, Elsevier COMCOM, and high-quality conferences, such as IEEE SECON, IEEE ICDCS, IEEE MASS, and EWSN. He published his first Springer book, "Challenges and Opportunities of Connected *k*-Covered Wireless Sensor Networks: From Sensor Deployment to Data Gathering" in August 2009. Also, he is the author and editor of two Springer books, "The Art of Wireless Sensor Networks: Fundamentals" and "The Art of Wireless Sensor Networks: Advanced Topics and Applications", which have been published in January 2014. In addition, he published these two current Springer books, "Mission-oriented sensor networks and systems: Art and science–Foundations" and "Mission-oriented sensor networks and systems: Art and science–Advances" in January 2019. He is the recipient of the US National Science Foundation (NSF) CAREER Award in January 2011, a 3-year US NSF Research Grant Award in June 2009, the National Security Agency (NSA) Award in 2017, and the Faculty Research and Development Grant Award from Hofstra College of Liberal Arts and Sciences in May 2009. In March 2014, he was recognized with the Distinguished Research Award at the University of Michigan-Dearborn. Furthermore, in May 2010, he was recognized with the Lawrence A. Stessin Prize for Outstanding Scholarly Publication (*i.e.,* Distinguished Research Award) at Hofstra University. He is the recipient of the Nortel Outstanding CSE Doctoral Dissertation Award in February 2009, and the John Steven Schuchman Award for 2006–2007 Outstanding Research by a Ph.D. student in February 2008. He received the Best Paper Award at EWSN in 2008, and the Best Paper Award at the IEEE PerCom 2008 Google Ph. D. Forum. He received several other prestigious awards, including the Best Graduate Student Paper Award (Nokia Budding Wireless Innovators Awards First Prize) in May 2004, the Best Graduate Student Presentation Award (Ericsson Award First Prize) in February 2004, and Laureate in Physics and Chemistry for academic years 1987 and 1988. Also, he was selected as the ACM Student Research Competition Finalist at the ACM MobiCom in September 2005. Also, he was selected for inclusion in the Marquis Who's Who in the World in

2019 and 2018, AcademicKeys Who's Who in Sciences Higher Education in 2017, Who's Who in America in 2017, AcademicKeys Who's Who in Engineering Higher Education in 2012, the AcademicKeys Who's Who in Sciences Higher Education in 2011, Feature Alumnus in the University of Texas at Arlington CSE Department's Newsletter in Spring 2011, Who's Who in America in 2010, and the 2008-2009 Honors Edition of Madison Who's Who Among Executives and Professionals. He received several service awards, including the Certificate of Appreciation Award at MiSeNet 2014, the Certificate of Appreciation Award at ACM MiSeNet 2013, the Certificate of Appreciation Award at the IEEE DCoSS 2013, the Certificate of Appreciation Award at the ACM MobiCom 2011, the Outstanding Leadership Award at the IEEE ICCCN 2011, and the Best Symposium Award at the IEEE IWCMC 2011. He serves as the Founding Coordinator of the CIS Distinguished Lecture Series, and as Coordinator of the CIS Faculty Research Talk Series since 2017. In addition, he was the Founding Coordinator of both the Distinguished Lecture Series and the Research Colloquium Series, in the College of Engineering and Computer Science at the University of Michigan-Dearborn from 2011 to 2015. He was successful to invite ACM Turing Award Winners to his distinguished lecture series, such as Dr. Manuel Blum from Carnegie Mellon University (CMU), and Dr. Shafi Goldwasser from MIT, who gave talks at the University of Michigan-Dearborn on January 25, 2013, and October 25, 2013, respectively, and Dr. Martin E. Hellman from Stanford University, who gave a talk at Fordham University on October 22, 2018. He was invited to give several invited talks at reputed universities. Indeed, he was invited to give a talk at the Third Arab-American Frontiers of Sensor Science Symposium, which was organized by the US National Academy of Sciences on December 5–7, 2015. Also, he served as external examiner of several Ph.D. Dissertations. He is the Founder of the Annual International Workshop on Mission-Oriented Wireless Sensor Networking (MiSeNet), which has been co-located with ACM MobiCom, IEEE INFOCOM, and IEEE MASS conferences since 2012. He served as Associate Editor of several prestigious journals, such as ACM TOSN, IEEE TC, IEEE Access, and Elsevier PMC. He serves

on the Steering Committee of MiSeNet, the Annual International Conference on Distributed Computing in Sensor Systems (DCOSS), and the International Workshop on Wireless Mesh and Ad-hoc Networking (WiMAN). Moreover, he served as General Chair, Program Chair, Track Chair, Session Chair, Publicity Chair, Web Chair, and Technical Program Committee member of numerous ACM and IEEE conferences, symposia, and workshops. He is an IEEE Senior Member.

Introduction

Habib M. Ammari

To raise new questions, new possibilities, to regard old problems from a new angle, requires creative imagination and marks real advance in science.

—Albert Einstein (1879–1955)

1 Mission-Oriented Sensor Networks and Systems: Art and Science

The fast advances in both inexpensive sensor technology and wireless communications over the last two decades have made the design and development of large-scale wireless sensor networks cost-effective and appealing to a wide range of mission-critical situations. These include area monitoring (e.g., deploying sensors for enemy intrusion detection, as well as geo-fencing of gas, oil pipelines, or work area), healthcare monitoring (e.g., using implanted, wearable, or environment-embedded sensors for medical applications), environmental/earth sensing (e.g., using sensors for monitoring air pollution and water quality, as well as detecting forest fire and landslide), industrial monitoring (e.g., deploying sensors for monitoring machine health, data center, data logging, water and wastewater, and structural health), to name a few.

Wireless sensor networking has attracted the attention of numerous practitioners and researchers from both industry and academia. These types of networks consist of a collection of tiny, resource-limited, low-reliable sensing devices that are randomly or deterministically deployed in a field of interest to monitor a physical phenomenon and report their results to a central gathering point, known as *sink* (or

H. M. Ammari (✉)
Wireless Sensor and Mobile Ad-hoc Network Applied Cryptography Engineering (WiSeMAN-ACE) Research Lab, Department of Electrical Engineering and Computer Science, Frank H. Dotterweich College of Engineering, Texas A&M University-Kingsville, Kingsville, TX, USA
e-mail: hammari@fordham.edu

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_1

base station). These tiny sensing devices suffer from their scarce capabilities, such as bandwidth, storage, CPU, battery power (or energy), sensing, and communication. In particular, the constrained power supplies of the sensors shorten their lifetime and make them unreliable. More precisely, mission-oriented sensor networks and systems are viewed as time-varying systems composed of autonomous (mobile) sensing devices (e.g., using mobile robots) that collaborate and coordinate distributedly to successfully accomplish complex real-time missions under uncertainty. The major challenge in the design of mission-oriented sensor networks and systems is due to their dynamic topology and architecture, which is caused mainly by sensing device mobility. The latter may have a significant impact on the performance of mission-oriented sensor networks and systems in terms of their sensing coverage and network connectivity. In such continuously dynamic environments, sensing devices should self-organize and move purposefully to accomplish any mission in their deployment field while extending the operational network lifetime. In particular, the design of mission-oriented sensor networks and systems should account for trade-offs between several attributes, such as energy consumption (due to mobility, sensing, and communication), reliability, fault tolerance, and delay.

Mission-oriented sensor networks and systems have been able to attract the attention of numerous people from scientific communities in both academia and industry. Indeed, a large number of related innovative research papers to solve challenging problems have been published in high-quality journals, conferences, and workshops. Given the importance of this area of research, I found it is essential that an up-to-date book on the above-mentioned topics be provided to our sensor networks and system research community. This book series, titled "*Mission-Oriented Sensor Networks and Systems: Art and Science*," includes two volumes, namely Volume 1 and Volume 2, whose titles are as follows, respectively:

- *Mission-Oriented Sensor Networks and Systems : Art and Science—Foundations*
- *Mission-Oriented Sensor Networks and Systems : Art and Science—Advances*

These two books have been assembled with a goal to address challenging and/or open research problems in traditional as well as new emerging areas of research in mission-oriented sensor networks and systems, including sensor networking, cyber-physical systems, and Internet of things, to name a few. It is worth mentioning that all the chapters in both volumes have been written as comprehensive review of the state of the art and state of the practice of their associated topics. Precisely, each chapter is either a survey of existing work in the literature or a survey with emphasis on the related research done by their corresponding authors. In either case, every chapter presents a thorough review of the underlying theoretical foundations, along with in-depth overview of the proposed approaches.

This book relates to the first volume, *i.e.*, *Mission-Oriented Sensor Networks and Systems: Art and Science–Foundations*. It aims to discuss topics in mission-oriented sensor networks and system research and practice, and provide the readers with opportunities to understand the major technical and application challenges of these types of networks, with respect to their architectures, protocols, algorithms, and application design. This book includes chapters that present novel theoretical and

practical ideas, which led to the development of solid foundations for the design, analysis, and implementation of energy-efficient, reliable, and secure mission-oriented sensor networks and system applications. Following Albert Einstein's above-quoted-wise approach to consider new and/or old yet challenging and/or open problems, all the chapters in this first volume focus on up-to-date research work that addresses a variety of problems in mission-oriented sensor networks and systems. In fact, this book covers various topics in mission-oriented sensor networks and systems, including sensor node architecture, sensor deployment, mobile coverage, mission assignment, detection, localization, tracking, data dissemination, data fusion, topology control, geometric routing, location privacy, secure communication, and cryptography. I believe that this book will be an excellent reference for graduate as well as senior undergraduate students who are majoring in computer science, computer engineering, electrical engineering, data science, information science, or any related discipline. Furthermore, this book will be a great source of information for computer scientists, researchers, and practitioners in academia and industry. I really hope that all readers will find this book very useful, nicely written, clear, exciting, and fascinating. My ultimate goal is that all users of this book will enjoy reading it and using it for any of their favorite research topics, as much as I enjoyed editing it.

2 Book Organization

This book consists of seven parts, each of which has two to four chapters. Next, we provide a short description of each part through a brief summary of each of its chapters.

In Part I, titled "*Architecture and Experimentation*," Chapter "Design Considerations of Mission-Oriented Sensor Node Architectures" presents some general considerations and architectures for sensor nodes. Then, it gives some insights of "how to design" the adequate node for specific use cases. Chapter "Failure Handling in RPL Implementations: An Experimental Qualitative Study" describes a standard for routing packets in low-power wireless networks, IPv6 Routing Protocol for Low-power and Lossy Networks (RPL). Then, it provides a performance evaluation of its two implementations, namely TinyRPL and ContikiRPL, in a range of link and node failure scenarios.

In Part II, titled "*Deployment and Coverage*," Chapter "On the Optimization of WSN Deployment for Sensing Physical Phenomena: Applications to Urban Air Pollution Monitoring" presents three formulations of the deployment issue of sensor and sink nodes based on integer linear programming (ILP) modeling while tackling the two main applications of air pollution monitoring. Then, it provides an analysis of the performance of the proposed models in terms of coverage and connectivity results through extensive simulations. Chapter "Mobile Coverage" reviews three categories of coverage, namely area coverage, barrier coverage, and sweep coverage, based on the sensor mobility. Then, it introduces various research problems for each category of coverage and critical issues caused by mobile sensors.

In Part III, titled "*Task Allocation and Mission Assignment*," Chapter "Energy-Aware Task Allocation in WSNs" focuses on the task allocation problem in wireless sensor networks. It provides an application-level taxonomy and an in-depth review of task allocation approaches. Chapter "Sensor Assignment to Missions: A Natural Language Knowledge-Based Approach" describes a knowledge-driven approach to intelligence, surveillance, and reconnaissance (ISR) asset assignment using ontologies, allocation algorithms, and a service-oriented architecture. Then, it analyzes the approach of using a representation based on Controlled English (CE) to improve the interface and human-in-the-loop aspects of the sensor assignment. Chapter " Resource Allocation and Task Scheduling in the Cloud of Sensors" discusses a new paradigm, called cloud of sensor (CoS), which faces challenges, such as the development of solutions for performing resource allocation and task scheduling in the CoS environment. Then, it gives an overview of the state of the art in the development of solutions for the above challenge.

In Part IV, titled "*Detection, Localization, and Tracking*," Chapter "Target Detection, Localization, and Tracking in Wireless Sensor Networks" investigates target detection approaches, sensor node localization algorithms, and target tracking schemes in wireless sensor networks. It shows that the integration of a plurality of homogeneous or heterogeneous sensors yields more accurate target detection, node localization, and target tracking, compared to the use of a single sensor. Chapter " Regularization-Based Location Fingerprinting" presents several ways on how to use regularization, which is a mathematical framework to learn a function from data by enforcing regularizers to improve generalizability. It shows how one can use regularization to learn from unlabeled fingerprints. Chapter "Sense-Through-Foliage Target Detection Based on UWB Radar Sensor Networks" studies sense-through-foliage target detection using ultra-wideband radars. It proposes a discrete cosine transform (DCT)-based approach for sense-through-foliage target detection, and a combined approach using radar sensor network and DCT, depending on the echo signal quality. Then, it applies these two algorithms to cognitive radar sensor network target detection. It uses a fuzzy logic system to automatic target detection based on the AC power values from DCT. Chapter "Mobile Target Tracking with Multiple Objectives in Wireless Sensor Networks" discusses a set of fully distributed tracking algorithms, which answer the query whether a target remains in a "specific area." Then, it proposes a tracking scheme, called t-tracking, to address the target tracking problem in wireless sensor networks, while considering several objectives, such as low capturing time, high energy efficiency, and high quality of tracking. Then, it validates the effectiveness of t-tracking using multiple objectives in extensive simulations and in a system implementation.

In Part V, titled "*Data Dissemination and Fusion*," Chapter "Data Dissemination and Remote Control in Wireless Sensor Networks" presents the challenges and research space of data dissemination and remote control in wireless sensor networks. Then, it reviews existing approaches, introduces relevant techniques, assesses various performance metrics, and compares representative methodologies. Also, it compares and elaborates on the existing approaches, namely structureless and structure-based approaches, depending on whether the network structure information is used during

the disseminating process. Chapter "A Data Fusion Algorithm for Multiple Applications in Wireless Sensor Networks" describes an algorithm, called Hephaestus, which is an information fusion distributed algorithm that uses an entropy procedure for data analysis of multiple applications in wireless sensor networks. Then, it presents the experiments for assessing Hephaestus in the context of a case study.

In Part VI, titled "*Topology Control and Routing*," Chapter "Underwater Networks for Ocean Monitoring: A New Challenge for Topology Control and Opportunistic Routing" reviews the peculiar characteristics of underwater wireless sensor networks and shows how knowledge acquired in terrestrial wireless sensor networks is impractical in underwater sensor networks. Then, it discusses intrinsic research challenges and provides some guidelines for the future design of topology control algorithms and opportunistic routing protocols for underwater sensor networks. Also, it gives some future research directions toward enabling large-scale deployments of underwater sensor networks for monitoring large areas of the ocean. Chapter "Geometric Routing Without Coordinates but Measurements" proposes alternative constructs to perform geometric routing over an efficient localization system, called virtual raw anchor coordinates. Then, it presents a geometric routing algorithm, where greedy routing and face routing are combined to guarantee the delivery of messages. Chapter "Delay-Tolerant Mobile Sensor Networks: Routing Challenges and Solutions" discusses the challenges for routing in the delay-tolerant mobile sensor networks, which can be terrestrial, underwater, or flying. Then, it presents a survey of existing routing algorithms in the literature, which are designed for delay-tolerant mobile sensor networks, delay-tolerant networks, or wireless sensor networks.

In Part VII, titled "*Privacy and Security*," Chapter "Location Privacy in Wireless Sensor Networks" analyzes potential threats in wireless sensor networks, highlights anonymity metrics, categorizes contemporary traffic analysis countermeasures, and discusses some of the emerging techniques. Chapter "Implementation of Secure Communications for Tactical Wireless Sensor Networks" presents an architectural framework for tactical wireless sensor networks by studying cybersecurity gaps and vulnerabilities within the 6LoWPAN security sublayer. Then, it discusses a key management scheme and a centralized routing mechanism that is non-broadcast but feasible in an operational scenario. Also, it tests the tactical wireless sensor network architecture against a variety of well-known network attacks. Chapter "Data-Driven Detection of Sensor-Hijacking Attacks on Electrocardiogram Sensors" describes a detector for identifying sensor-hijacking attacks that alter sensor readings in a wearable medical Internet of things. It shows that the proposed temporal electrocardiogram alteration detector has promising results. Chapter "Cryptography in WSNs" provides a concise review of cryptography used in wireless sensor networks.

3 Acknowledgments

This complete two-volume series book, titled "*Mission-Oriented Sensor Networks and Systems: Art and Science*," is a tribute to the outstanding work of the foremost

leading authorities and scholars in their fields of research in the area of mission-oriented sensor networks and systems. Honestly, it is unfair that my name only appears on the book cover. And, it is really a great pleasure and an honor for me to cordially recognize all of those who contributed a lot to this book and generously supported me throughout this project in order to make this two-volume series a reality. Therefore, it is really a great privilege for me to work with all of these talented scholars. Without them, it would not be possible at all to finish this book and make it available to all the researchers and practitioners, who are interested in the foundations of mission-oriented sensor networks and systems.

First and foremost, I am sincerely and permanently grateful to Allah—the Most Gracious, the Most Merciful—for everything He has been giving me. In particular, I would very much love to thank Him for providing me the golden opportunity to work with such group of outstanding scientists and researchers to put together this book and for helping me publish it within three years. I am extremely happy and so excited to dedicate this modest book to Him and very much hope that He would kindly accept it and put His Blessings in it. His Saying "**And of knowledge, you (mankind) have been given only a little**" has an endless, pleasant echo in my heart and always reminds me that our knowledge is much less than a drop in the ocean.

It is worth mentioning that all the contributing authors were invited to contribute to this book, and that no Call for Book Chapters had ever been sent out through any mailing list. All of those authors whom I invited were chosen very selectively to cover most of the foundational topics in mission-oriented sensor networks and systems. They have been contributing to the growth and development of the field of mission-oriented sensor networks and systems. This book would never have been written without their great contributions, support, and cooperation. Thus, my cordial recognition is due to all of my friends and colleagues—the ones whom I invited to contribute with their chapters to this book—whose names are listed in the alphabetical order: Drs. Gabriel Aquino, Nikolaos Baroutis, Walid Bechkit, Md Zakirul Alam Bhuiyan, Jorge Blasco, Azzedine Boukerche, Eyuphan Bulut, Hang Cai, Thomas M. Chen, David W. Courtney, Flávia C. Delicato, Alberto Garcia-Ortiz, Yuan He, Konrad Iwanicki, Hyunbum Kim, Ulf Kulau, Pierre Leone, Hao Liang, Jing Liang, Qilian Liang, Claudio Miceli, Agnieszka Paszkowska, Harsh Kupwade Patil, Paulo F. Pires, Luci Pirmez, Alun Preece, Stephan Rottmann, Kasun Samarasinghe, Igor L. dos Santos, Preetha Thulasiraman, Duc A. Tran, Krishna K. Venkatasubramanian, Gary M. Weiss, Lars Wolf, Wanli Yu, Mohamed Younis, Xiaolong Zheng, and Albert Y. Zomaya. I am really honored to have worked with such an amazing crew of scholars and scientists. I learned a lot from them throughout this project, and it was an incredible experience for me in finishing this book.

Each chapter has undergone two rounds of reviews. Moreover, in each round, every chapter received 2–5 reviews by experts in the scope of the chapter. Our ultimate goal is to provide the readers with a high-quality reference on the foundations of mission-oriented sensor networks and systems. Precisely, all chapters were carefully reviewed in both rounds by all the contributing authors. I would like to express my sincere gratitude to all the contributing authors for their constructive feedback to improve the organization and content of all chapters. My special thanks go to

Drs. Damian M. Lyons (external reviewer), Flavia Delicato, and Mohamed Younis for their generous offer to review several chapters of both books of this two-volume series. Also, my original plan was to publish only one book, titled "*Mission-Oriented Sensor Networks and Systems: Art and Science*." But, I ended up with 42 chapters, which I split into two volumes along with their chapters and titles. Moreover, I am very grateful to Dr. Thomas F. La Porta, IEEE Fellow; Evan Pugh, Professor; and William E. Leonhard, Professor, Department of Computer Science and Engineering, and Director of the School of Electrical Engineering and Computer Science at Penn State University, for their great foreword, kindness, and endless support to me.

I started this project on Monday, September 5, 2016, at 12:42 AM when I contacted Publishing Editor, Dr. Thomas Ditzinger, who approved my proposal for an edited book. All chapters for both volumes as well as the two forewords were uploaded on the Web site of Springer and made accessible to Project Coordinator, Mr. Gowrishankar Ayyasamy, on August 21, 2019. Hence, this project lasted about three years. During all this period of time, I exchanged a few thousands of emails with all contributing authors with regard to their chapters. I would like to thank all the contributing authors for their invaluable time, flexibility, and wonderful patience in responding to all of my emails in a timely manner. Please forgive me for your time, and I hope that the readers will appreciate all of your great efforts and love all the materials in this book. We all have devoted a considerable amount of time to finish this book, and I hope that all of our efforts will be paid off in future.

I would like to acknowledge all of my family members who have provided me with an excellent source of support and constant encouragement over the course of this project. First of all, I am extremely grateful to both of my first teachers, my mother, Mbarka, and my father, Mokhtar, for their sincere prayers, love, support, and encouragement, and for always teaching me and reminding me of the value of knowledge and the importance of family. I owe them a lot and cannot find my words to thank them enough for everything they have done to make me who I am now. Also, I am most grateful to my best friend and beloved wife, Fadhila, for her genuine friendship and for being extremely supportive and unboundedly patient while I was working on this book. In addition, I would like to express my hearty gratitude to my lovely and beautiful children, Leena, Muath, Mohamed-Eyed, Lama, and Maitham, for their endless love, support, and encouragement. They have been one of my greatest joys, very patient, and understanding. I hope they will forgive me for spending several hours away from them while I was setting in front of my PC in my office or my laptop at home busy with this book. Several times, they all told me: "Daddy, as usual, your books and emails are always dragging you away from us!" My lovely wife and children have been a wonderful inspiration to me and very patient throughout the life of this project. Without their warm love and care, this project would never even have been started. Furthermore, my special thanks and gratitude go to all of my sisters, brother, nieces, and nephews for their love, thoughtful prayers, concern, and valuable support all the time.

This project could not have been completed without the great support of the people around me who made this experience successful and more than enjoyable. I would like to thank all of my friends and colleagues at Texas A&M University-

Kingsville and, particularly, my fellows in the Department of Electrical Engineering and Computer Science, for the collegial and very friendly atmosphere they provided me with to finish this book. In particular, I am very grateful to my Dean, Dr. Mohammad Alam, Fellow IEEE, IET, OSA, SPIE, IoP, IS&T, and IAPR, Professor of Electrical Engineering, in the Department of Electrical Engineering and Computer Science, Frank H. Dotterweich College of Engineering, at Texas A&M University-Kingsville, for his kindness, continuous encouragement, and outstanding support to WiSeMAN-ACE Research Lab since I joined the Department of Electrical Engineering and Computer Science at Texas A&M University-Kingsville in August 2019. Also, I am very thankful to Dr. Afzel Noore, Professor and Associate Dean for Undergraduate Affairs in the Frank H. Dotterweich College of Engineering at Texas A&M University-Kingsville, for his humbleness and outstanding support to me in several ways. Furthermore, I would like to express my profound gratitude to all of my colleagues, including faculty and staff members, in the Department of Electrical Engineering and Computer Science at Texas A&M University-Kingsville, especially Drs. Rajab Challoo, Reza Nekovei, Syed Iqbal Omar, Sung-won Park, Scott Smith (EECS Department Chair), Lifford Mclauchlan, Mais Nijim, Amit Verma, Muhittin Yilmaz, Nuri Yilmazer, Muhammad Aurangzeb, Gahangir Hossain, and Maleq Khan; and Mrs. Debra Beltran and Mr. G.R. Benavides, for their extended support and encouragement. Moreover, I would like to convey my warm thanks and appreciation to all the faculty and staff members in the Frank H. Dotterweich College of Engineering Dean's Office who have been so helpful and very kind to me, namely Drs. Mahesh Hosur, Associate Dean for Graduate Affairs and Research; Robert Diersing, Executive Director, High-Performance Computing Center; Tamara Denise Guillen; Julissa Flores; Rosenda Garcia; and Rose Anna Gomez. Also, I would like to convey my special thanks and deep appreciation to Dr. Zakaria Abd-Elmageed, Professor in the College of Pharmacy at Texas A&M University, College Station, for his friendship, kindness, and outstanding support to me in several ways. This work is partially supported by the National Science Foundation (NSF) grants 0917089 and 1054935.

Last but not least, I would like to express my deep appreciation and gratitude to Dr. Thomas Ditzinger, Publishing Editor; Mr. Gowrishankar Ayyasamy and Ms. Janet Sterritt-Brunner, Project Coordinators for Books Production; Ms. Daniela Brandt, Project Coordinator for Publishing; Ms. Sabine Gutfleisch, Editorial Assistant; Mr. Sooryadeepth Jayakrishnan, Project Manager for Book Production; and Ms. Sylvia Schneider, Production Coordinator for Books Production. It was a great pleasure to work with all of them. I would like to acknowledge the publisher, Springer, for the professionalism, patience, and the high quality of their typesetting team as well as their timely publication of this book.

August 21, 2019

Part I
Architecture and Experimentation

Design Considerations of Mission-Oriented Sensor Node Architectures

Felix Büsching, Keno Garlichs, Ulf Kulau, Stephan Rottmann and Lars Wolf

Abstract In research and education, a Wireless Sensor Network (WSN) may exist for its own sake and also the specific nodes used in such networks might be considered as study objects. However, in real applications the networks and the nodes are applied to solve real-world issues and, hence, have to be designed for their specific purpose. Since WSNs and according nodes have to cope with significant limitations and challenges, especially regarding energy budgets, it is typically considered as impractical to use a 'one size fits all' network configuration or a 'one size fits all', universal sensor node. Instead, it is necessary for every single component of a node, like processor, memory, radio transceiver, set of sensors, peripherals, and energy source to consider what is necessary and they have to be chosen according to the needs of the envisaged use case. Therefore, designing or at least selecting appropriate nodes is a crucial part for every deployment of WSNs. Based on that also the used networking technologies, the topology, the protocols, etc. have to be developed and chosen. In this chapter, first some general considerations and architectures for sensor nodes are presented. Also, some insights of'how to design' the adequate node for specific use cases are given. The design of sensor nodes for two exemplary missions are discussed in detail. In particular the diverse missions *Human Activity Monitoring* and *Smart Farming* are used to reveal the specialty when designing mission-oriented sensor nodes.

F. Büsching · K. Garlichs · U. Kulau · S. Rottmann · L. Wolf (✉)
Institute of Operating Systems and Computer Networks, TU Braunschweig,
Mühlenpfordtstraße 23, 38106 Braunschweig, Germany
e-mail: wolf@ibr.cs.tu-bs.de

F. Büsching
e-mail: buesching@ibr.cs.tu-bs.de

K. Garlichs
e-mail: garlichs@ibr.cs.tu-bs.de

U. Kulau
e-mail: kulau@ibr.cs.tu-bs.de

S. Rottmann
e-mail: rottmann@ibr.cs.tu-bs.de

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_2

1 Challenges for Sensor Nodes

The field of applications for WSNs is massive and use cases are nearly limitless. They vary from wildfire detection in remote forests (e.g. [1, 2]) to real-time capable sensor-/actuator networks for mobile robots in factory automation scenarios (e.g. [3]). The respective requirements are mostly different. In the first case it is sufficient to occasionally sense temperature and humidity and communicate only if necessary, i.e. when a possible wildfire was detected. An external power source cannot reasonably be provided since the nodes will be deployed in remote places, far apart from each other. Therefore, the wireless sensor nodes have to be powered by independent means like batteries, solar panels or similar. Due to the location of the nodes and their sheer number, they have to last as long as possible without human interaction. As a result, the most crucial requirement to the nodes is energy efficiency. Computational power on the other hand is rather negligible.

The second case has quite contradictory requirements. The nodes are attached to large robots which are likely to be consuming several orders of magnitude more energy than the most powerful classical wireless sensor node. Energy can unhesitatingly be drawn from the machines power supply and as a result, energy efficiency is of much less importance compared to the first case. This fact is very important because the presented use case demands a high throughput for the network and the computational units. In factory automation scenarios, data often has to be sensed precisely and with high sample rates. That data has to be evaluated and decisions have to be made based upon it to allow fast reactions. At least some of those steps are likely not to happen on the same node and, therefore, data has to be distributed through the WSN. The application often demands reaction times in the order of milliseconds. This can only be achieved with high precision time synchronization and real-time capable communication technologies [4]. There are different protocols which can be used for the synchronization is this case: e.g. the Precision Time Protocol (PTP), standardized as IEEE 1588 [5] or other approaches research came up with, e.g., [6–8]. In order to support the usage of such protocols and use-cases, the requirements regarding computational power of the nodes as well as wireless transmission rates are rather high.

In many WSN scenarios, sensors like accelerometers or gyroscopes are not required whereas they surely are inevitable in other areas, like Ambient Assisted Living (AAL), which evaluate human micro movement. Nodes in remote regions have to be tamper proof since they cannot be frequently checked for manipulation. Nodes worn on the body, like in AAL, should rather be lightweight and small to allow for a comfortable wearing.

When teaching about WSNs in schools and universities, the focus might be on the ease of programming code for the nodes and flashing them. A requirement could be e.g. to support modern WSN operating systems like Contiki OS [9], RIOT OS [10] or FreeRTOS [11]. A USB connector and a bootloader allowing to flash the nodes will be beneficial as well. Besides that, the nodes do not need to be specifically designed for one purpose. More general ones might be suited better to support a variety of

student projects. The cost should be low, but compared to mass deployments, it is not that important—allowing to equip the nodes with lots of different sensors.

Ultimately, design requirements are different in every use-case and often they are even competing (e.g., computational power vs. low energy consumption, light weight vs. large battery). While for educational purposes, more general nodes might be preferable, most real-world applications demand highly specialized, mission-oriented wireless sensor nodes. As a result, many node architectures have been developed and are available on the market. Still, the "one size fits all" node is not existent and will probably never be, as specialized hardware may be needed for every use case.

1.1 Outline

The remainder of this chapter is structured as follows: In Sect. 2, important general design considerations are proposed and exemplary, the popular and legendary wireless sensor node TMote Sky is introduced. Afterwards, in Sect. 3, the previously mentioned AAL project "Design of Environments for Ageing" is presented and used as exemplary use case for Sect. 4 to show how a mission-oriented wireless sensor node was designed for that specific use case. The design process along with the production as well as the final evaluation of the resulting node are shown there. As a second exemplary mission, Smart Farming is introduced in Sect. 5. It is used to showcase another mission-oriented sensor node—Amphisbaena, which is additionally discussed in Sect. 6. Finally, Sect. 7 concludes this chapter.

2 General Node Architecture

When planning a mission-oriented WSN, the abundance of different requirements leads to the fact that a node is specialized for its desired application. Hence, there is a plethora of existing nodes which differ in terms of computational performance, wireless capabilities, energy demands and especially the set of sensors or actuators. However, the architecture of a wireless sensor node can be cut down to a general architecture which is a common guideline when designing a specialized node.

2.1 Components of a Sensor Node

A high-level view of the general architecture valid for most sensor nodes is shown in Fig. 1. Usually, every sensor node consists of the components shown in the figure, indeed, with individual variations.

Power Unit/Power Supply: One of the most important considerations for the design of the node is the power supply. The thing to determine is which energy source will be available to the nodes in the respective use case. Several options are possible such as energy from a wall socket, (replaceable) batteries, or intermittently available

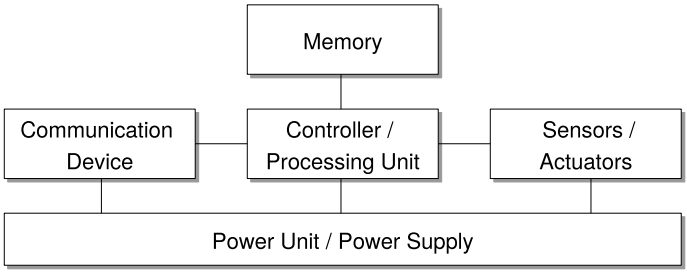


Fig. 1 General architecture of a sensor node

power produced by energy harvesting (e.g., solar panels). When generating electrical energy at the node, a buffer is required, which might be a battery or capacitor.

For battery powered systems, one has to consider for which amount of time the node needs to be able to run. Critical factors may be the weight and size or how often the node can be accessed to change the batteries. All other design decisions may be limited by the amount of energy available: Computational power, radio transmission range, sensors and things like elements for user interaction (such as displays or LEDs).

Controller/Processing Unit: An essential part of the sensor node to decide on is the Micro Controller Unit (MCU) which runs the actual application. Several things have to be considered. An important aspect is the computational power of the MCU. If collected data has to be processed on the node itself, more power may be needed. Traditionally, 8 bit or 16 bit controllers have been used, while in the last years more low power 32 bit devices have emerged on the market. Using a higher clock rate means faster processing at the cost of increasing energy demands. A trade-off between processing power, energy demand and costs has to be found for the application.

Interface buses: As in most WSN applications, the sensor node needs to measure physical quantities like temperature, voltages or pressure, for example, sensors are needed. Since they are usually not implemented into the MCU itself, external chips have to be connected. For connecting sensors to the controller, usually standardized bus systems will be used. It often makes sense to agree on very few interfaces that all the peripherals to be attached can support. For wireless sensor nodes, this will mostly include Inter-Integrated Circuit (I^2C) well as Serial Peripheral Interface Bus (SPI) since almost every sensor or radio chip is available with at least one of those interface buses and all widely used MCU platforms support them, too. The same considerations can and should be applied to actuators and other peripherals like memory or storage extensions.

Sensors/Actuators: Suitable sensors have to be found that can provide the desired level of accuracy and precision while at the same time not exceeding the energy budget. As mentioned before, the current consumption is important for the sensors. Some sensors may even require a higher operating voltage than the controller itself. If this case occurs, additional components like step-up converters may be an option.

Sensors with an analog interface require an analog-digital converter (ADC). One might choose the internal converter of the MCU (if present), or an external, which again may be connected via I^2C or SPI. In both cases, a reliable and stable voltage reference may be required, depending on the accuracy needed in the actual application.

Memory: When dealing with sensor nodes, different kinds of memory can be found. The program itself is usually stored in the integrated flash memory of the MCU. The amount of program memory needed depends on the application running on the node. The same is true for the RAM. As the RAM is volatile (and very limited), and the internal flash is to be programmed externally, measured data is usually stored in other kinds of memory. External flash memory chips exist with a capacity in the magnitude of a few megabytes. Another option for data storage can be SD cards which are rather cheap offering up to several gigabytes of capacity. Recent MCUs may have an SDIO interface for communicating with the cards, or they may be connected to the SPI bus. EEPROM memory integrated into the MCU is often used for keeping configuration data in a volatile memory.

Communication Device: The same considerations as before have to be made when selecting a radio for the sensor node or the complete network. Here, the parameters are achievable throughput, communication range, available frequencies and energy consumption. In most cases, frequencies in the ISM band will be chosen, which are available (mostly) all over the world. But still, care has to be taken for some bands, which are only allowed to be used in some regions, while they may not be available in others. In many cases, communication technologies like IEEE 802.15.4 or Bluetooth are used. IEEE 802.11 standards offer higher bandwidth but require more energy. The LoRa [12] technology offers a very long communication range for low power radios, at the cost of an extremely low data rate compared to other mentioned standards. While proprietary radio modules and protocols are available, in some cases it is more beneficial to use standardized technologies to simplify the interconnection with other devices. For example, Bluetooth and WiFi is available at virtually any smartphone or laptop computer.

Antenna: For transmitting and receiving radio signals an appropriate antenna is mandatory. First of all, the antenna must fit to the desired frequency range of the radio transceiver. Additionally, the directivity of the antenna has to be taken into account. An ideal isotropic antenna would have an uniform three-dimensional radiation pattern and a perfect 360° horizontal and vertical beamwidth—unfortunately it only exists in theory. In reality, antennas are often classified as directional and omnidirectional. Directional antennas focus the transmitted energy in a certain direction; thus, a high gain can be achieved. Omnidirectional antennas provide a 360° radiation pattern—but, other than the term “omni” might imply, only in one plane. Thus, an omnidirectional antenna does not necessarily imply universal receive or transmission characteristics. Moreover, the location of the antenna has to be considered. It can either be part of the PCB which makes it cheap, but space consuming; or, a chip antenna can be soldered on the PCB, which is a little more expensive, but typically smaller in dimension. An external antenna can be a good choice if for some reason shielded (e.g. Ex-compliant) housings are needed.

2.1.1 Systems on a Chip and Development Boards

In terms of physical size, complete System-on-a-Chips SoCs may be a good choice. They combine the components (often except sensors) mentioned before in a single chip. An example for a cheap solution with a powerful 32 bit controller and WiFi as well as Bluetooth is the ESP32 SoC [13], an IEEE 802.15.4 radio combined with an 8051-based MCU which can be found in the CC2531 [14] series by Texas Instruments or Atmega256RFR2 [15] series by Atmel/Microchip.

Especially when it comes to these SoCs, the question arises, what actually *is* a sensor node. Development boards by chip manufacturers may offer an MCU combined with sensors and/or a radio. However, those boards usually have a large form factor and are not optimized in concerns of energy demand.

2.2 General Example: TMote Sky/TelosB Sensor Node

When looking for "the standard" wireless sensor node, probably one will find the TMote Sky, respectively TelosB, developed at Berkeley University of California [16]. This section provides a brief overview on this commercially available node that has been widely used in many research projects. *Google Scholar* lists 4.290 publications that mention TMote Sky, respectively 7.820 mentioning TelosB[1] which underlines the popularity of this type of WSN node. Although this node was introduced more than 10 years ago in 2005, the TMote Sky/TelosB and its clones (and variants) like the Maxfor MTM-CM500-MSP[2] are still used in recent Wireless Sensor Network (WSN) deployments and testbeds [17–19]. The TMote Sky was intentionally designed to be used in academic research rather than for specific applications with distinct requirements. However, its architecture follows the design rules of the general node architecture as described in the previous Sect. 2. A Block diagram of the TMote Sky's architecture is depicted in Fig. 2.

The processing unit consists of a Texas Instrument MSP430F1611 [20] micro-controller which integrates a 16 bit RISC micro-processor, memory (48 kB flash, 10 kB RAM) as well as typical peripherals such as Universal Synchronous/Asynchronous Receiver/Transmitter (USART), I^2C, SPI, analog-digital converter (ADC), General Purpose Input/Outputs (GPIOs) and timers.

The radio transceiver—a CC2420 [21]—is connected to the processing unit via SPI. The CC2420 is an IEEE 802.15.4 compliant packet based radio transceiver and therefore well suited to be used in Wireless Personal Area Networks WPANs like WSNs. It operates within the unlicensed ISM band at 2.4 GHz and allows a maximum data rate of 250 kBit/s.

To store information on a non-volatile memory, the TMote Sky is equipped with a 1 MBit flash which is also connected to the SPI.

[1] As from May, 2017.

[2] https://www.advanticsys.com/shop/mtmcm5000msp-p-14.html.

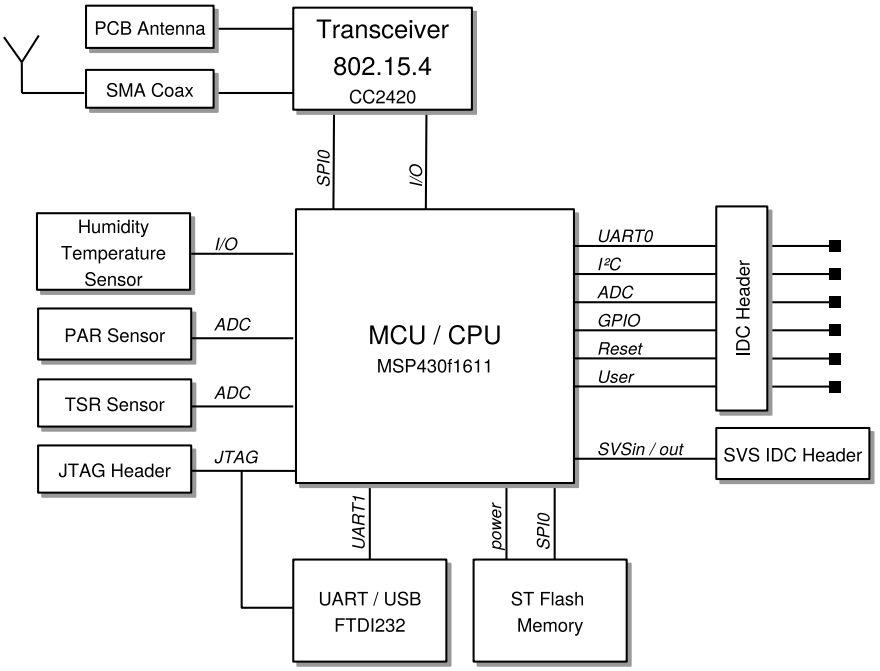


Fig. 2 Block diagram of TMote sky sensor node

Although no specific application was intended, the TMote Sky includes a sensor set for exemplary sensing applications. A temperature and humidity sensor, a Photosynthetically Active Radiation Sensor (PAR), and a Total Solar Radiation Sensor (TSR) are available on the node. In addition, buses and GPIOs are connected to a pin-header so that further components can be interfaced by the TMote Sky.

Besides two AA-batteries (mounted on the rear side), the USB-port can be used as a power source. Furthermore, USB makes programming easy because no additional hardware is needed as the TMote Sky supports bootstrap loader (BSL) [22].

The comprehensive support of the WSN operating systems Contiki OS [9] and TinyOS [23] are another reason why the TMote Sky has firmly established within the WSN community.

Nevertheless, with the evolution of WSNs towards the ever-growing Internet of Things (IOT) the complexity and requirements of protocols and applications has increased. As an example, our own experiences show some of the limitations of this node: During the GINSENG project [24] we had to deal with the insufficient amount of addressable memory of the TMote Sky, respectively the MSP430-architecture. Due to the lack of program memory we were not able to run the developed MAC protocol “GinMAC” combined with the support for IPv6 [25] at the same time.

In summary, the TMote Sky is still a valuable node when basic characteristics of WSNs are investigated but when it comes to recent and mission oriented WSNs, it is

common practice to design a specialized node whose design is strongly defined by its particular application.

3 Exemplary Mission: Human Activity Monitoring

After generally having described the considerations to make when designing a typical wireless sensor node in Sect. 2 and introducing the most popular, commercially available and general platform TMote Sky in Sect. 2.2, we now want to provide a detailed case example how to design a mission oriented wireless sensor node. In particular the following Sections will introduce the process from analyzing the requirements, designing and building to evaluating and improving a wireless sensor node for human activity monitoring.

3.1 Motivation and Requirements Analysis

Within the Project "Design of Environments for Ageing" [26], it was planned to monitor the activities of elderly people and by this to perform a fall detection and a fall prevention through gait analysis. For the field test, it was targeted to equip more than 30 persons with sensor nodes. The intention was that the persons should wear the nodes for most time of the day.

For human activity monitoring, top priority is to have the right set of sensors. In most cases an accelerometer is used (e.g. in [27, 28]); newer studies also benefit from a gyroscope [29] and a pressure sensor [30]. So, the data of these sensors should be recorded simultaneously at a data rate of at least 50 Hz.

Secondly, power consumption is a major issue, as a long-term monitoring is envisaged and the changing of batteries is unacceptable for the monitored persons. Additionally, the batteries should be rechargeable as some energy demanding monitoring tasks should only be performed during the daytime; in this case, nodes should easily be recharged during the night.

Size and weight of a sensor node is the third aspect to be addressed when choosing or designing a node which is intended to be worn for longer times.

Whenever there is no continuous radio link to a sink and constant transmission of the recorded data cannot be guaranteed, sufficient (non-volatile) memory is needed to—temporarily or persistently—store the recorded data from various sensors. Some of the elderly people to monitor had no Internet connection at their homes, so, it was planned to store the movement data on the nodes. Every week the elderly person or the supervisor should replace a memory card with a fresh one and send the used card via mail to the scientists performing the data analysis.

A fifth issue is the total cost as there was only little money planned to equip and perform the long-term monitoring

Thus, the *ideal* node had to fulfill the following requirements:

1. Sensor set consisting of accelerometer, gyroscope, and barometer (pressure sensor).
2. Low power consumption; rechargeable batteries.
3. Wearable and robust.
4. Storage (non-volatile memory)—exchangeable.
5. Cheap—not more than 100 Euro for each device.

So, in a first approach, it might be a plan to analyze if any of the (commercially) available nodes could fulfill all of the requirements.

3.2 Market Analysis

As presented and discussed in the previous section, one of the most frequently used Wireless Sensor Nodes in research or education is the TelosB [31]/TMote Sky [16] / MTM-CM5000-MSP/... It is a quite universal node, easy to handle, supported by operating systems like Contiki OS [9] or TinyOS [23], but—due to it's age—limited in computational power and memory capacity. Many similar nodes, based on TI's MSP-430 MCU, exist, for instance, the Shimmer Sensor [32] for human activity monitoring which will be discussed below. The sensors of TMote Sky only measure light, humidity, and temperature. Thus, if we used the TMote Sky as a basis, we would have to add further needed sensors—maybe via a daughter-board. But, besides the lack of appropriate sensors there are many other aspects which argue against using the TMote Sky as basis, especially the energy supply and the memory capacity.

3.2.1 Shimmer Sensor

The Shimmer Sensor [32] is widely used in the area of human activity monitoring. Its basic design is close to the TMote Sky as it is another MSP430-based sensor node, but there are some mentionable differences (Fig. 3). First of all, it comes with an accelerometer which is attached to the MSP430 MCU. Secondly, a slot for a microSD card is present which can be used as data storage for long term monitoring. While earlier versions (by design) only supported the use of either the IEEE 802.15.4 radio or the microSD card, later revisions do not have this problem anymore. It normally runs TinyOS [23] which makes it easy to program.

In its basic version, with just an accelerometer present, the Shimmer node costs already about 200 Euro. There are several extensions on PCB available, that can be mounted on the basis module. A gyroscope module costs another 150 Euro and a pressure sensor can only be obtained when purchasing a GPS daughter-board for 150 Euro. As the Shimmer sensor has a proprietary connector, an additional charging and programming module for 200 Euro is needed. For our designated set of sensors consisting of accelerometer, gyroscope and pressure sensor, the whole

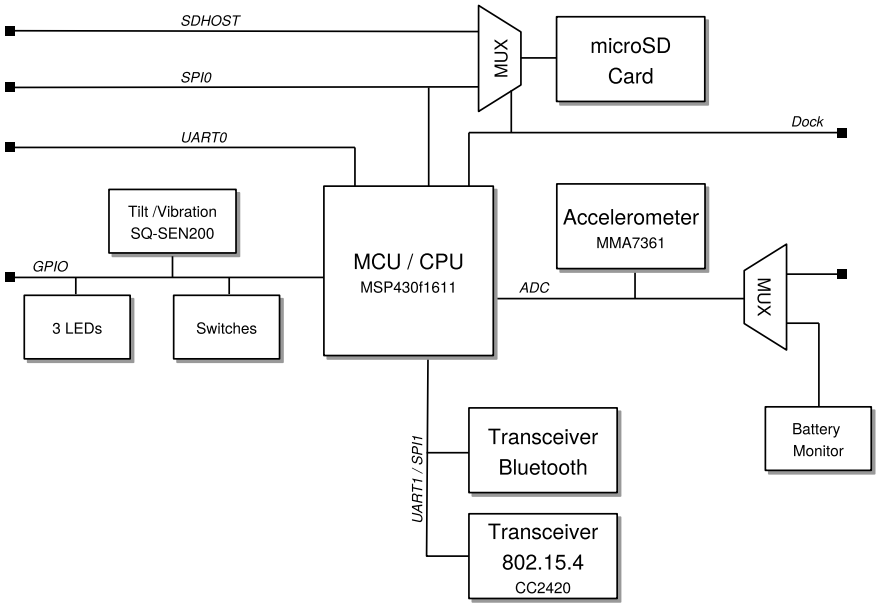


Fig. 3 Shimmer architecture

platform would cost up to 700 Euro and then becomes very bulky due to the two daughter-boards.

3.2.2 Medical and Commercial Products

Especially for activity monitoring, there are several commercial and medical products available. The MTI Actigraph and the Sensewear Pro II armband monitor [33] are just two examples for wearable long term (offline) monitoring devices. Newer versions are additionally equipped with a proprietary radio transceiver. Both have in common that they come with relatively huge (256 MB or more) flash memory for storing long term monitoring data. However, they also share the attribute of being "closed" solutions with fixed software installed for a very specialized use case. Detailed specifications, as well as source code, are not available and, thus, own applications are (if at all) not easy to implement and deploy.

More nodes, platforms and architectures exist, but, to the best of our knowledge, none fulfilled all of our requirements, concerning functionality, attached sensors, size and price.

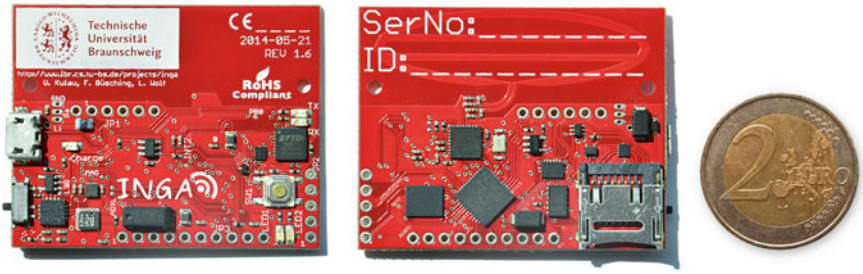


Fig. 4 INGA's front and back view, nearly in original size and compared to a 2-Euro coin. INGA's physical dimensions are 50 mm × 39 mm × 7 mm

4 Mission Oriented Sensor Node: INGA

As explained earlier, several wireless sensor nodes for many different purposes are available on the market already. But, none of them suited the specific challenges imposed by the targeted application of human activity monitoring [26] when we needed them for a specific project. Hence, the following case example of the INGA [34] sensor node (c.f. Fig. 4) illustrates how a wireless node is designed, implemented and evaluated towards its desired application.

4.1 Design Decisions

The decisions for INGA's design, functionality and equipment were derived from the targeted application (cf. Sect. 3) and inspired by pros, respectively cons, of existing sensor node platforms (cf. Sects. 1, 2.2 and 3.2). As the "Design of Environments for Ageing" [26] is a research project, the specific design decisions go beyond the functional requirements for activity monitoring. The usability, the costs and expandability are also of importance when composing the node.

Starting with the processing unit, an MCU fitting all our requirements like energy efficiency, simplicity, usability and a full open source toolchain support had to be selected. For this reason, only simple 8/16-bit MCUs were considered because better performing architectures (e.g., 32 bit ARM processors) are less practical for WSN beginners that should use the node within the research project. The reduced complexity of less powerful devices is more suitable as also freshmen undergraduate students should be able to develop hardware drivers and applications to support the project.

Figure 5 shows INGA's overall hardware architecture. Some of the characteristics are detailed in the following subsections. The center of the design is the Atmel ATmega 1284p MCU together with an Atmel AT86RF233 IEEE 802.15.4 compliant 2.4 GHz transceiver. All communication buses are separated, which leads to a high level of robustness because a malfunctioning or falsely programmed device only

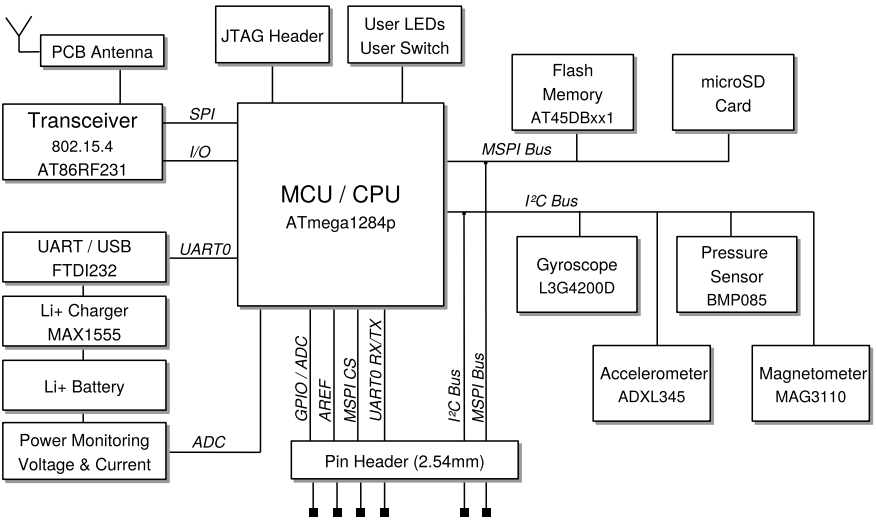


Fig. 5 Block diagram of INGA's architecture

affects the bus it is attached to and not the whole system. Furthermore, all relevant buses, unused I/O channels, and other useful signals are led through to a 2.54 mm pin header to allow extensions.

4.1.1 Processing Unit: MSP430 Versus ATmega

With regard to the software support of existing WSN nodes, it seems rational to choose between Atmel ATmega and TI MSP430 MCU architectures. The instant support of recent operating systems like Contiki OS [9] or TinyOS [23] are a beneficial feature of these MCUs. As mentioned above, for some applications the MSP430F161 of the TMote Sky lacks of addressable memory, but there are updated MSP430 MCUs which provide a more suitable program memory size. The MSP430 is a 16bit RISC MCU and is widely used in the area of wireless sensor networks, as opposed to the ATmega, which is a 8 bit RISC architecture and not that widely spread in that specific area of WSNs. Nevertheless, the ATmega MCUs are supported by a huge community.

Both MCUs are available with a comprehensive set of peripherals so that it is worth looking at the details of an MSP430F161 and an ATmega1284p, respectively: While the MSP430F161 has two USART components, which can either be configured as SPI, I^2C or USART, the ATmega1284p has also two USART but additionally separated I^2C and SPI interfaces. This is crucial because one USART in any case has to be configured as USART for serial communication to a PC. The remaining USART in the MSP430 has to change the protocol (SPI, I^2C) during operation. Considering an extensive usage of motion sensors, the permanent reconfiguration of the interface

leads to an overhead in time and, thus prunes the performance and efficiency of the entire node.

Especially when the node is used for research the reprogramming capabilities are of importance. A node with a special programmer (AVR Raven [35]) or a docking station (Shimmer [32]) is inappropriate and too cost-intensive and impractical in terms of remote programming. Thus, programming via a standardized USB interface is the first choice. The additional opportunity of wireless over-the-air (OTA) programming would be a benefit for the future. For this purpose the ATmega1284p offers a bootloader section within its flash memory which is well suited to implement USB and OTA programming.

Balancing the pros and cons we selected the ATmega1284p MCU to be used as processing unit.

4.1.2 Communication Unit: AT86RF233—Radio Transceiver

The Atmel AT86RF233 is a fully IEEE 802.15.4 compliant radio transceiver within the unlicensed ISM 2.4 GHz band. Thus, it is compatible with the majority of common sensor nodes used in WSNs. With regard to the transmission of (privacy) sensitive data, which is expected by the given application, this radio offers AES hardware en-/decryption support [36]. A unique feature of this radio is the Phase Difference Measurement Unit (PMU) which can be further used to implement an indoor localization based on the principle of active reflector [37].

To reduce costs we decided to use a PCB antenna instead of an external or chip-antenna. The actual PCB antenna is designed as a folded dipole with an impedance of about 100 Ω and derived from the original Atmel Application Note [38]. The radiation pattern and the gain of up to 6.5 dBi were evaluated in previous simulations and underpin the selection of this antenna.

4.1.3 Sensing Unit: Focused on Human Activity Monitoring

To allow a monitoring of human activities, INGA is equipped with a comprehensive set of Microelectromechanical Systems (MEMS) sensors. The sensors are directly connected to the dedicated I^2C-bus, shown in Fig. 6. For further usage it is also led through on pin headers. An easy expansion is, e.g., realizable by an I^2C IO expander that allows the connection of multiple additional inputs and outputs.

Accelerometer—Analog Devices ADXL345: In prior investigations [39] we compared various different 3-axis accelerometers in terms of power consumption, linearity, bit noise and correlation. It turned out that the ADXL345 was most qualified for our purposes. Its sensitivity can be set to ±2, ±4, ±8 and ±16 g; the sampling resolution varies from 10 to 13 bit (dependent on sensitivity) and it has an adjustable sampling rate of up to 3.2 kHz.

Gyroscope—ST Microelectronics L3G4200D: The ST-Microelectronics L3G-4200 MEMS gyroscope is able to measure deviations of orientation and, thus, to

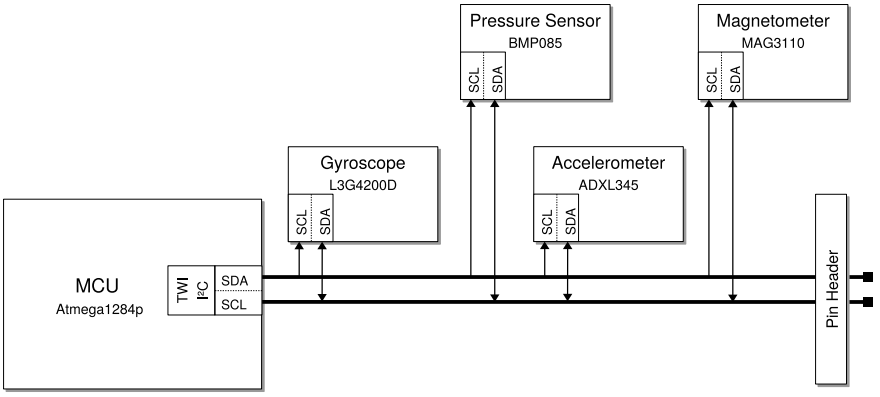


Fig. 6 INGA's I^2C-Bus bus and attached components: accelerometer, gyroscope, magnetometer and pressure sensor (temperature sensors are included)

determine the location more precisely. It is able to detect up to 2000 degrees per second in three axes (16 bit). It extends the sensor node by the measurement of 3 additional degrees of freedom. In addition, the gyroscope has an integrated temperature sensor but with limited resolution (8 bit).

Magnetometer—MAG3110: Regardless to the orientation of the INGA, the MAG3110 3-axis magnetometer is able to implement an electronic compass. With a resolution of 10 μT and a full scale range of ± 1000 μT the MAG3110 can, e.g., measure the terrestrial magnetic field. Together with the accelerometer and the gyroscope INGA allows the measurement of 9 degrees of freedom.

Pressure Sensor—Bosch BMP180: The pressure sensor is able to sense pressure with a resolution of 0.01 hPa and an accuracy of ± 0.12 hPa. This allows the detection of a difference in altitude in the dimension of few centimeters. Thus, with regard to gait monitoring, it can be decided easily whether a person walks up- or downstairs. The pressure sensor has 16–19 bit resolution, depending on the selected sensitivity. Also another temperature sensor (16 bit) is integrated which enables the sensor to do a temperature compensation and, by this, to measure absolute pressure.

4.1.4 Memory Devices: Storage of Measured Data

Especially when recording data of human activities for research purposes a high resolution of the measurement variable as well as a high sample rate is required. To prevent that every single data item has to be transmitted separately and soon via the transceiver unit, INGA is equipped with two non-volatile memory devices. A serial flash device is a convenient way for fast data storage while an microSD card is ideal to store mass of data. Both memory devices are connected to an MSPI-bus which is realized through the second USART of the ATmega1284p. The benefit of a second SPI is that communication with other SPI devices does not interfere the

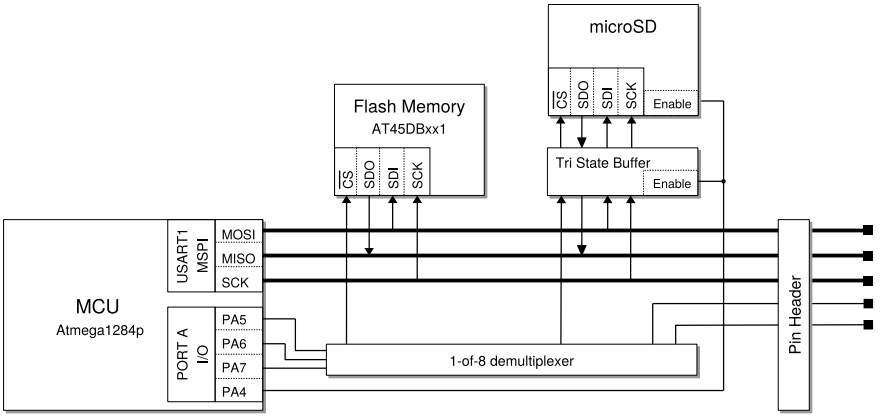


Fig. 7 INGA's MSPI bus and attached components: accelerometer, flash memory, microSD card

communication with the radio transceiver. Figure 7 illustrates that three I/O ports are used to demultiplex the chip-select signals of the attached devices; this way up to seven devices can share the MSPI bus. Two of these chip-select lines are led through a pin header for individual expansions.

Serial Flash—Atmel AT45DBxx1 Serie: INGA can be equipped with either one of AT45DB081 (8 MBit), AT45DB161 (16 MBit) or AT45DB321 (32 MBit) serial flash devices. The dual buffer interface of these devices leads to a significant speedup in contrast to single buffer devices because one buffer is still capable of communication with SPI while the other writes/reads the flash memory. The benefits of this device are explained in greater detail in the Evaluation Section.

microSD Card: The SD card specification is not standardized in ISO or DIN and is only available for paying license holders. Luckily, there are open protocols that allow a free but slow operation of any SD card via SPI, but there are some peculiarities to deal with. First of all, SD cards can be very power consuming with a current of up to 45 mA during operation. There is no easy way to just switch off the supply power because SD cards can draw current from the data lines as well due to their internal design. Another major issue appears when attaching more than just the SD card to an SPI, because the specific protocol requires some action on the clock line (SCK), without chip-select being enabled. As this happens on a bus, undefined states can occur resulting in communication problems on the whole bus. This problem is solved by introducing a tri-state-buffer that is able to "disconnect" all lines of the SD-card and by this there is nearly no power consumption of the SD-card while not in use.

4.1.5 Power Unit: Management and Monitoring

When connected to USB, an attached Li+-battery can be directly charged through a MAX1555 Li+-battery charger. The voltage of the attached batteries is constantly monitored by a potential divider and the current consumption of the system can be monitored by the combination of a Maxim MAX4372F high-side current-sense amplifier with voltage output and a shunt. Both, the voltage and the current monitor are connected to an ADC-channel of the ATmega1284p. A Maxim MAX 8881 low-dropout linear regulator provides a constant voltage of 3.3 V, which corresponds to the minimum voltage level due to the SD card support.

4.1.6 USB Interface

A universal serial connection for communication with a PC and programming the device is realized by the widespread FTDI232R UART-USB converter attached to the first USART. It is supported by every PC operating system without the need of installing special hardware drivers. It also acts as power source when connected to a hub or PC.

Programming: The USB interface is also used for flashing INGA's firmware. Firstly the FTDI232R is used to implement a rudimental In System Programmer (ISP) by using the bit-banging mode [40]. In case of the ATmega1284p the ISP is based on an SPI protocol which allows uploading data to the program memory of INGA.

However, a more flexible way to program the node is the usage of a bootloader. INGA's Bootloader also allows flashing via USB and is compatible to AVRDUDE,[3] thus, no additional hardware is required for flashing INGA. With enhanced drivers (provided for Linux, Mac and Windows by FTDI), also the capability of resetting the MCU is implemented. This allows the flashing of multiple connected nodes quasi simultaneously. By adjusting the clock rate on demand we were able to speed up the bootloader's transfer rate by the factor of 6 compared to the default clock rate of 8 MHz. This way, e.g., flashing and verifying Contiki OS binaries (40 kB) via USB is done in less than 5 s. The advantage of a bootloader is the starting point for an OTA flashing which will be implemented in software in the near future: With a bootloader present it is regardless on which memory the operating system to boot is stored. Thus, it has just to be taken care of the secure and accurate wireless transfer of the operating system, the rest can be handled by the bootloader.

[3] http://www.nongnu.org/avrdude/.

4.2 Evaluation

Extensive evaluations have been done to verify the compliance to the originally posted requirements. The evaluation of INGA in real world environments took place in our lab with other 2.4 GHz hardware present and in the countryside with most likely no other radio traffic in the considered frequency spectrum. INGA's default clock rate is 8 MHz, but, to be comparable to the well known TMote Sky node (cf. Sect. 2.2) which normally runs at 4 MHz, measurements of the application layer throughput were performed at both 4 and 8 MHz for INGA.

4.2.1 Communication Range

In a first evaluation, we compared INGA's communication range with the original Atmel AVR Raven node as it uses the same type of antenna (cf. Sect. 4.1). It is important to support the monitored person to be, e.g., in the garden while still being able to report a fall event. In a line-of-sight setting on a field with no other interfering radio transmissions in the designated frequency, the UDP/IP packet loss at increasing distances were measured. The tested nodes acted as sender, sending 6 B of payload every 20 ms and a PC with an IEEE 802.15.4 AVR USB-stick[4] acted as receiver. The transmission power of both devices were set to 3 dBm.

It turned out that there is no significant difference between INGA and AVR Raven as both nodes had only randomly occurring single packet losses along the track. We defined a UDP packet loss of greater than 50% as breakpoint where no further communication is possible. This breakpoint was reached after 194 m for INGA and 219 m for AVR Raven. The increase of packet loss happened in a short period of only few meters from nearly zero percent to greater than 50%. Thus, INGA's radio-frequency (RF) part is fully working and comparable to the AVR Raven.

4.2.2 Antenna Characteristics

As described above, INGA is equipped with a high gain PCB antenna. To complement the evaluation of INGA's RF section, the antenna characteristic of the horizontal plane was measured. It is important to assure reliable communication regardless of the node's orientation which cannot be controlled when worn by a person in its everyday life. Hence, potential shadowing or the ideal radiation angle is ascertained for this sensor node design.

The evaluation was done with two INGA nodes at a distance of 15 m. One node rotated around z-axis and send 6 B of payload data via RIME [41] to a sink node periodically. The transceiver unit of INGA (Atmel AT86RF233, cf. Sect. 4.1) offers the opportunity to read-out the energy level of received packages. The energy detection (ED) measurement is done by averaging the RSSI value over eight symbols, which

[4] http://www.atmel.com/tools/rzusbstick.aspx.

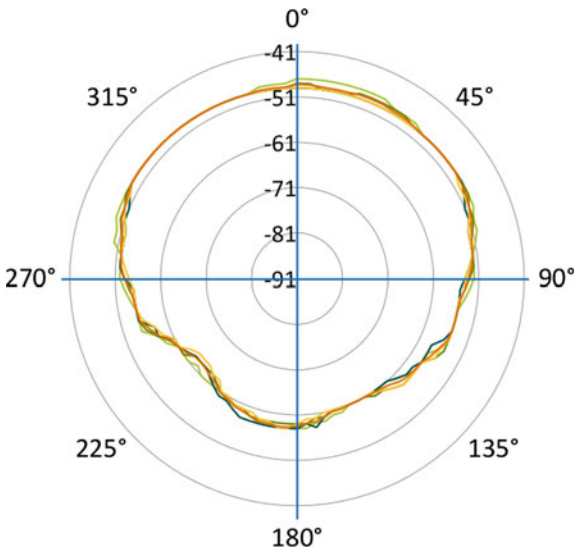


Fig. 8 Measured antenna characteristics (horizontal plane) in dBm

is done by the transceiver of the sink node itself. Afterwards the RF input power can be calculated manually through the following equation:

$$P_{RF} = -91 + ED\,[dBm] \tag{1}$$

Figure 8 shows the results of five rotations. The back part (from 90 to 270°) is more shadowed cause of the PCB design. However, no blind spots could be detected which would effect the wireless communication.

4.2.3 Application Layer Throughput

In our lab we compared the UDP throughput of INGA to the TMote Sky nodes using the UDP/IP and the RIME [41] communication stack of Contiki OS. Packets of varying payloads were sent in each case between two identical nodes which were placed in a distance of one meter. We measured the exact time for 100 packets with an oscilloscope by toggling GPIOs. In Fig. 9 the UDP throughput is plotted for different payloads. INGA's throughput is higher at any payload size. A maximum throughput of 129224 bit per second (16.153 KByte/s) was achieved for 90 B payload size by INGA running at 8 MHz. Even when INGA's system clock is set to 4 MHz it still outperforms the TMote Sky.

In Fig. 10 the RIME throughput is plotted for different payloads. The overall throughput of RIME is higher. INGA outperforms the TMote Sky here too, at least with bigger payloads. A maximum throughput of 155874 bit per second (19.484 Kbyte/s) was achieved for 90 byte payload size by INGA running at 8 MHz. At

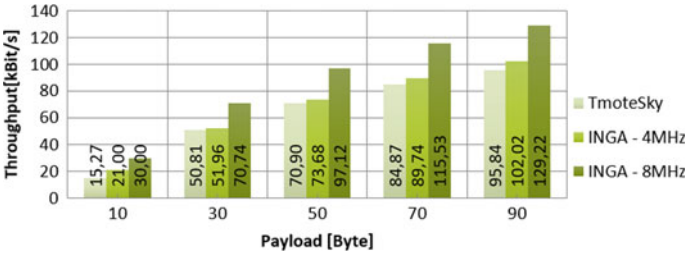


Fig. 9 The throughput of UDP/IP traffic at a varying payload size

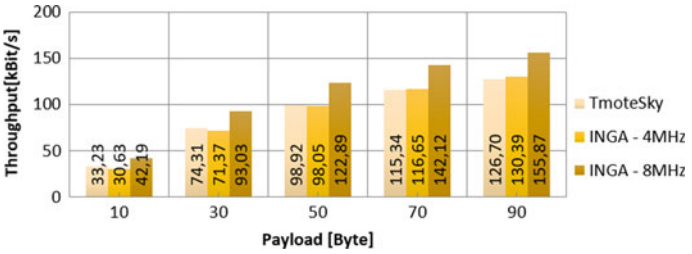


Fig. 10 The throughput of RIME traffic at a varying payload size

small payload sizes the TMote Sky seems to perform better than INGA running at 4 MHz, but at a high price as will be shown below.

4.2.4 Memory Performance

In contrast to the MSP430-based architectures, the ATmega based architecture of INGA has separate interfaces for memory and radio access. To expose this feature we evaluated a simple but common scenario were other nodes begin to fail: UDP traffic of increasing throughput shall be received and then be written into the external flash memory. Using common nodes like the TMote Sky one would expect an increase of packet loss or a decrease of throughput because writing data is time consuming and at some throughput the MCU is busy writing data. In contrast, INGA's dual-buffer flash (cf. Sect. 4.1) in combination with the designated second SPI was able to write any received packet directly to flash without any losses. Figure 11 illustrates INGA's performance in receiving data packets and writing them to external flash memory at varying rates of throughput in comparison to the TMote Sky. The TMote Sky shows first stirrings of packet loss at a throughput rate of 90 kbit/s. As TMote Sky itself was not able to send packets at such high data rates, we used another INGA as sender.

When SD cards are connected to low-power MCUs, the slow SPI mode for communication is used instead of the more powerful proprietary interface. To demonstrate the effect of this slow interface, received data was directly written to the SD card. In Fig. 12 these results are shown. It can be seen that at a higher rate of throughput,

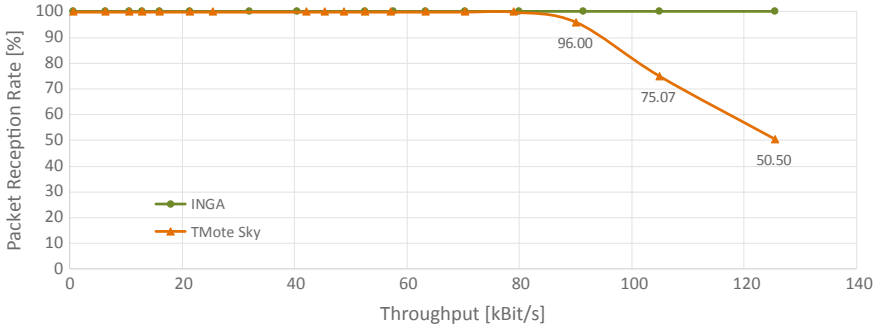


Fig. 11 Performance of writing received UDP data directly to external flash memory

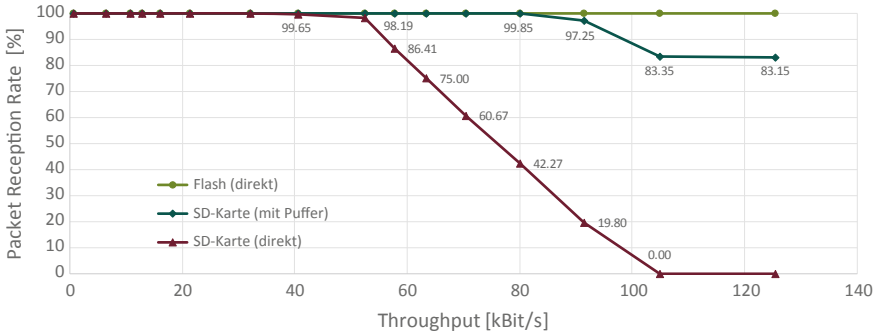


Fig. 12 INGA's performance of writing received UDP data to flash memory and to SD card (with and w/o SRAM buffer)

packets get lost and thus the SD card limits the throughput of this scenario. An obvious optimization is to introduce an SRAM buffer and with that to write whole pages instead of every single packet to the SD card which reduces the control overhead significantly. It can be seen, that with such a buffer, still data rates of more than 80 Kbit/s are possible, whereas without such a buffer, a maximum of 40 Kbit/s can be reached.

In case of expected burst traffic it would also be a suggestion to first write all received data into the external flash and copy from there to the SD card afterwards. Copying one flash page from external flash to SD card takes 24 ms; the throughput from flash to SD card is 21.33 Kbyte/s.

4.2.5 Power Consumption

Power consumption is crucial in WSNs and also for the desired use case where nodes should be attached to humans. The node will run battery powered and the energy consumption thus has to be as small as possible. Nevertheless, the actual power

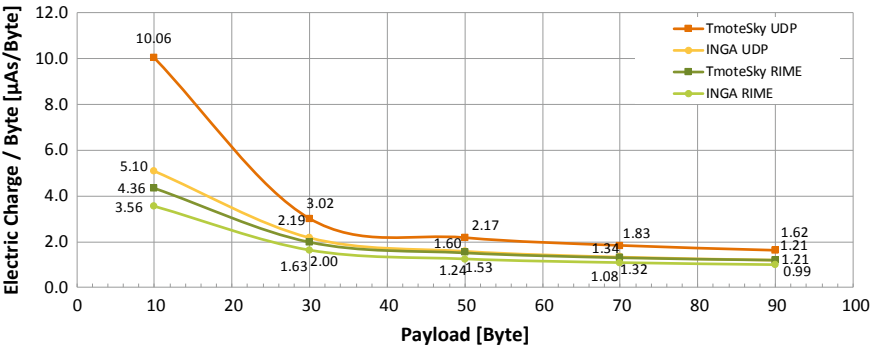


Fig. 13 The required electrical charge to transmit one byte for different transport protocols and different payloads

consumption of a node or a network depends on the use case and the tasks the nodes and the network in general are performing. INGA has the capability of online current and voltage monitoring. The MCU, the transceiver, the sensors and the memories all have different power saving states leading to numerous possible evaluations of power consumption. In a small setup, INGA's overall power consumption was measured in comparison to the TMote Sky. Figure 13 shows the electrical charge required to transmit one byte of payload data at different payload sizes and different transport protocols. At small payload sizes, INGA's dominance is obvious, but also at high payload sizes the TMote Sky still consumes at least 20% more energy to transfer data wirelessly.

A more detailed view on INGA's power consumption is given in [42] where we used a modified version of INGA to implement and evaluate Dynamic Voltage Scaling (DVS) techniques.

4.2.6 Cost-Efficiency

As mentioned in the design decisions, INGA was designed to be cost efficient. INGA is fully open source so that the basic layout can be modified to fit individual needs. As all information is public available, it is sufficient to send gerber files of the layout and a Bill of materials BOM to PCB manufactures which offer the full service of PCB production; they will take care of ordering the parts and assembling the whole PCB. To give an example, our first two prototypes cost €318 each, which is by no means cost efficient. However, when ordering a bigger amount of nodes the costs drop significantly. All costs given in Table 1 include material, components, PCB, taxes and shipping for Germany, when producing INGA.

Table 1 Prices per INGA

Pieces	10	50	100	1000
Price/Piece	€111.60	€61.17	€51.52	€30.04

4.2.7 Resources

As already mentioned, INGA is completely open source. You are free to adapt or change anything you like. We provide schematics, EAGLE-files and Gerber-files in the download section of INGA's website. So, INGA can be downloaded, build and used as it is or easily adapted to special needs (e.g. additional sensors) as every design tool is free available. In addition, a Contiki OS version with full support of INGA is provided in a Github-repository. Some hardware add-ons, like a LED-bar, a Bluetooth transceiver, and a GPS shield were already developed and will be made available soon. Additionally, some teaching materials and tutorials are provided at the project page.[5]

4.3 Limitations and Future Work

INGA's set of sensors seems to be specialized for human activity monitoring, but it is not limited to this. The same sensor set can, e.g., be used for flight control of quadrocopters. For many experiments such as the field tests with more than 30 persons "wearing" INGA, enclosures are needed. We designed such and can produce them with a 3D-printer. INGA is also supported by Contiki's simulator Cooja [43], which is also a useful lecture when developing software on the desk. Nevertheless, the simulation performance is limited as AVRORA [44] is used to emulate the Atmel1284p MCU and the AT86RF233 radio transceiver which implements the entire instruction set in Java. In sum, simulations of a double-digit number of nodes are possible but the simulation of hundreds of nodes is impractical at the moment.

5 Exemplary Mission: Smart Farming

Related to the previous mission of human activity monitoring, the following Sections will introduce the process from analyzing the requirements, designing and building to evaluating and improving a wireless sensor node for smart farming applications.

Smart farming, or Precision agriculture, is a fast growing application area for WSNs where the features of these networks can help to optimize the agricultural

[5] http://www.ibr.cs.tu-bs.de/projects/inga.

yield by selective usage of fertilizers, controlled sprinkling or early detection of plant deceases. With the primary intention of WSNs to allow distributed sensing applications even without any existing infrastructure, the usage of WSNs on agricultural areas is almost ideal.

5.1 Motivation and Requirements Analysis

The regular method to monitor the health of plants are sparse spot tests by a farmer and the individual know-how to suggest the health of the entire plant population. With regard to the size of agricultural areas, these spot-tests are less precise as the characteristic of the soil, e.g. nutrient content or water retention, is highly heterogeneous. At this point WSNs are ideal as they can be deployed throughout the entire agricultural area while spanning up a communication network without any existing infrastructure to forward sensed data to points of interests, e.g. farm machines, nodes with GSM connection. Thus, an area-wide monitoring can be achieved to optimize farming processes even further—keyword: Cyber Physical System (CPS).

In recent years there have been a few experimental WSN deployments related to smart farming [45–48]. However, the experiences so far revealed that nodes and networks deployed on an agricultural area have to face several challenges and the operation of WSNs for smart farming is not trivial.

Energy Efficiency: WSNs that are deployed on agricultural areas have to work autonomously for a long lifetime of several months (regular farming season) or even years (e.g. nursery gardens). Considering the limited capacities of e.g. batteries lead to the fact that energy is one of the scarcest resources in such WSNs. A common approach to prolong the lifetime of nodes is energy harvesting [49]. Nevertheless, it depends on the particular application whether electrical energy can be derived from external sources. For example the usage of solar power is not always possible as solar panels might be covered by plants or dirt which prunes the efficiency significantly.

Limited Maintainability: Due to the size and the fact that agricultural areas are located in rural areas, nodes deployed on a field are hard to access and the maintainability is limited. Thus, with regard to the aforementioned energy demand, it is impractical to exchange batteries of nodes to prolong the lifetime of the WSN. Moreover, while a single node is not of importance for the entire distributed sensing, it hast to be guaranteed that the desired application is not affected by failing nodes. Similar to the batteries, an exchange or repair of nodes is associated with an disproportional overhead. For this reason a key-challenge for WSNs in smart farming applications is the dependability.

Harsh Environmental Conditions: Nodes deployed on a field are directly exposed to challenging weather conditions like rain, direct sunlight or fog. In the past the WSN community gained a lot of experience with WSNs for outdoor applications e.g. [50, 51]. One aspect is that nodes are often exposed to extreme temperatures [52, 53]. The nodes suffer from extreme temperatures as the efficiency of

commonly used IEEE 802.15.4 transceivers decreases with growing temperatures and thus, the reliability of transmissions decreases as well [54, 55].

Beside the meteorologic impacts, the plants themselves affect the characteristics of the WSN. In [45, 46] it was observed, that the reliability of links decreases with growing plants. The water-rich plants shadow the communication and might lead to a break-down of links and thus changing topologies with potential partitioning of the network.

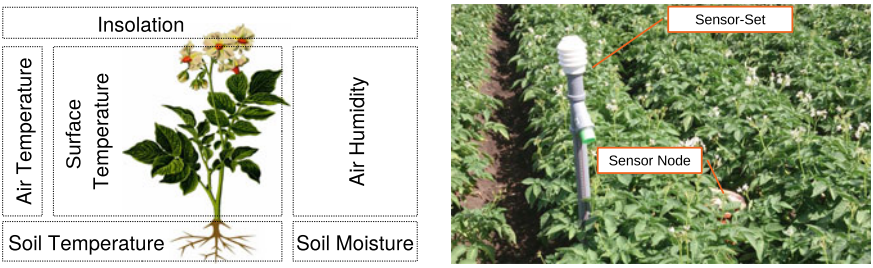
Limited Connectivity: As mentioned in the previous paragraph, even the intra-communication—the communication between the nodes within the WSN—is limited. In addition, also the inter-communication—the forwarding of collected data to a dedicated sink, e.g. a cloud server—is challenging. A cellular network is not always available and especially in rural areas, where WSNs in smart farming applications are usually located, the network operators do not put so much effort in network expansion. A promising technology might be LoRa [12], but as LoRa only provides low data-rates for long-range communication, which is not sufficient to be used for sole data uplink. Instead, Delay Tolerant Networks DTNs can be applied to deal with broken links or failing routes to a sink. By using farming machines or drones, data can be used as data mules to collect aggregated data from the field to the cloud.

5.2 Exemplary Smart Farming Scenario

In this chapter, we will deal with the design of network components for an exemplary smart farming application to monitor the plant health. In particular we will consider an application tackling the distributed measurement of the crop water stress index [56] of potato plants. As the term *stress* in connection with plants might be unfamiliar, the following paragraph will provide some background information:

Plants are stressed, when growing conditions are not optimal such as the absence of water or inferior soil. With regard to the climatic change this effect becomes a serious issue as agricultural areas begin to silt and the soil water retention decreases. Thus, with detailed information about the condition of plants, the usage of sprinkling and fertilizers can be optimized. In cooperation with a potato research station we deployed nodes with a dedicated sensor-set able to measure plant stress influencing factors on a potato field. The rating of the plant's condition is based on several parameters. Besides soil moisture, soil temperature, air temperature and humidity, an important parameter is the surface temperature of the plants which can be measured non-invasively by a simple infra-red temperature sensor. Whenever a plant is *stressed* it will immediately stop evaporation and the plant heats up which can be revealed by this sensor (Fig. 14).

In our exemplary scenario several sensing nodes are deployed on a field to enable an area-wide measurement of the parameters to determine the crop water stress. A schematic overview of the network's architecture is shown in Fig. 15. Here, we can find different classes of the nodes, ranging from rather small *WSN nodes (DCN)*



(a) Parameters for crop water stress index

(b) Deployment of a prototype on a potato field

Fig. 14 Measurement of the crop water stress as an exemplary smart farming application for WSNs

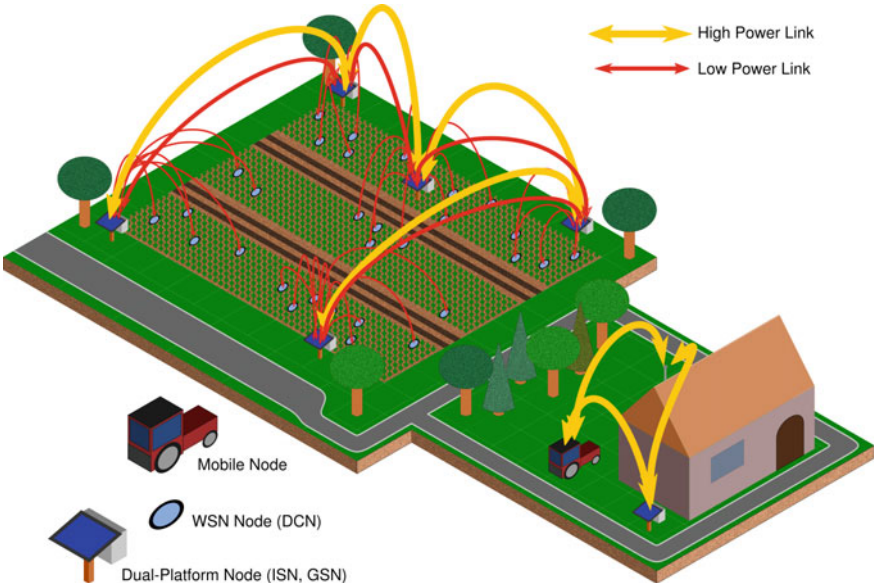


Fig. 15 Smart farming example: network layout including WSN nodes and dual-platform nodes

to embedded *Dual-Platform Nodes (ISN, GSN)* running a full-featured operating system like Linux.

Data Collection Node (DCN): Those nodes can be referred as "typical" sensor nodes to measure specific parameters as described before. In this use case, many of them would be deployed on a field in order to be able to get fine-grained measurements of dimensions. Large distances and the vast number of nodes make it unfeasible to route data of all nodes to a single sink node since the nodes closer to the destination would have to transfer many data sets resulting in a short battery life.

Intermediate Sink Node (ISN): To avoid depleting batteries and congestion of the radio links, a second kind of nodes is introduced. These nodes are stationary with

the ability to charge batteries using photovoltaic panel and are equipped with a large storage. In order to be able to forward received data efficiently to a *Global Sink*, these nodes consist of two platforms. A *low power platform* runs on a 32 bit MCU and has a low power radio for standards such as IEEE 802.15.4 available, while the *high power platform* is a Linux-running Single Board Computer SBC. The latter one has a WiFi radio which enables a fast transfer of large amounts of data to other nodes.

Global Sink Node (GSN): The last class of nodes are in terms of hardware the same or very similar to the *Intermediate Sink Nodes*. The difference is that they act as a gateway to the final destination for the data. They may transfer the data to mobile data mules passing the field, or they can be equipped with a Wireless Wide Area Network WWAN radio offering a connection to the Internet.

Each DCN is assigned a corresponding ISN, where it will send its data to. For this data transfer, different routing protocols and network architectures can be taken into consideration. Data received on the ISN's low power platform can collected in its memory, which is a SD card in our case.

5.2.1 Sensing Tasks of Dual-Platform Nodes

Besides acting as a sink for the surrounding WSN only, the dual-platform nodes can be used to collect, data, too. Here, the system can benefit from the architecture, as well. Considering a sensing pattern as shown in Fig. 16, the low power platform can be equipped with low power sensors as found on any conventional WSN node (temperature, humidity, …), while the high power part of the node can control more complex sensors, such as cameras.

The Fig. 16 shows a typical sensing scenario, in which sensors have to be read out in a given interval. The temperature might have to be sampled every few minutes. If we had only a simple time switch which would turn on the SBC for each measurement, it would have to be booted very often, as given as "SBC only" in the figure. Since both platforms are closely coupled, temperature readings can be conducted by the low power platform and the results either be sent using the low power radio, or handed out to the SBC when it is booted the next time. The SBC needs to be booted only when an image with the camera has to be taken or large amounts of data need to be

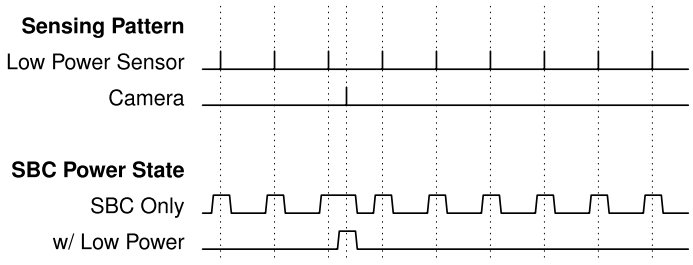


Fig. 16 Exemplary sensing patterns of dual-platform node

Fig. 17 Photo of a deployed node with several sensors attached

transmitted, which cannot be done by the MCU-based board with its radio. As can be seen, the up-time of the SBC can be reduced significantly with the approach of a dual-platform architecture.

6 Mission Oriented Sensor Node: Amphisbaena

To handle a scenario as described in Sect. 5.2, we implemented the dual-platform node Amphisbaena [57] to be used as a ISN or GSN. Since only few of these nodes will be deployed, they can be mounted with exposition to direct sunlight above the crop, or at the edge of a field and thus can make use of energy harvesting with photovoltaic panels. A photo of a node deployed on a potato field can be seen in Fig. 17.

6.1 Collaborative Data Collection

The low power platform of the ISNs and the GSNs is available for communication with its radio virtually at any time, while the high power part, the SBC can be booted on demand. Small sensor nodes for data collection can transmit their measurements to the low power part of the dual-platform node, where they can be stored on the SD card. Taking into account that many nodes transfer their data to each sink node, it becomes clear that it is unfeasible to send *all* data via the low power nodes to the GSN. Thus, the DCNs send their data to the corresponding ISN. As mentioned before, the ISN may also take measurements with different kinds of sensors and be turned on for this task. In our case, the protocol for data transmission is DTN.

When a certain amount of data has been collected at the ISN, the high power platform can be turned on while neighboring ISNs can be asked to do the same. So, a network with a high data capacity will be established, depicted by the orange arrows in Fig. 15. In this manner, the complete path of any ISN to the GSN could be set up at once, while in some cases it might be enough to forward the data only to the next hop, from where it will be forwarded later. This method of forwarding data with the store-and-forward approach is supported by the use of DTN, in this network, too, since some nodes might not come up as desired due to power failures or similar.

Since the high power platform of the GSN might also run from batteries, it needs to save energy as well and thus the uplink may also be powered down. If the uplink is based on a mobile node such as a tractor, it may happen that the meeting time of the nodes is not known in advance. The contact duration may be rather short since the mobile node is moving.

The actual protocol used for communication depends strongly on the scenario, network layout and architecture for the application. For the system described in this chapter, DTN is used as an example.

6.2 Design Considerations

The ISNs and GSNs used in the example deployment [58] consist of two platforms—a low power board as well as a powerful, Linux based SBC. The low power board is based on a STM32F407 MCU which offers communication interfaces such as USB or Ethernet besides aforementioned buses like SPI or I^2C. A detailed overview of the components is given in Fig. 18.

Basically, similar design considerations as for the *INGA* node have been taken for the low power platform, while the field of application is quite different. For the development of Amphisbaena, more processing capabilities and additional interfaces like USB and Ethernet are required, and thus, the 32 bit MCU has been chosen. Since this part of the network is to be deployed statically with a photovoltaic panel and a large battery, this choice leading to a higher current draw is reasonable.

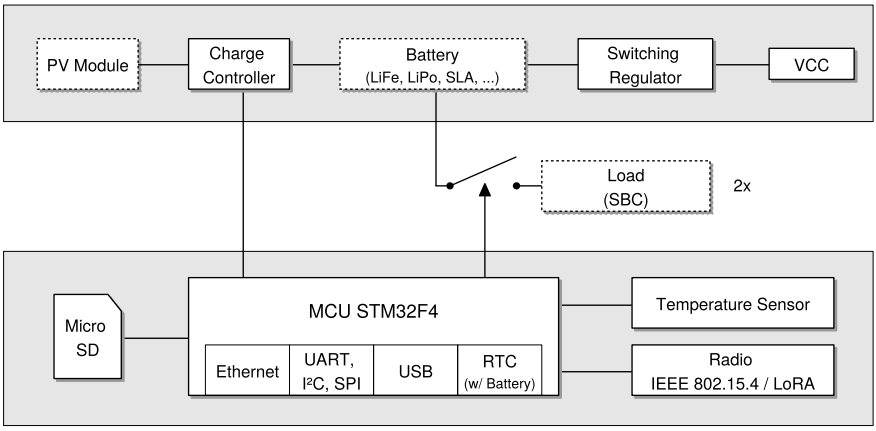


Fig. 18 Components of solar-powered dual-platform node

MCU: Besides interfaces for several buses and General Purpose Input/Output (GPIO) pins, the STM32F4 MCU features a battery-backed Real-Time Clock (RTC) to keep the time. It is running on *FreeRTOS* [11]. The MCU can be clocked up to 168 MHz.

Temperature: A 1-wire temperature sensor *DS18B20* can be mounted on the Printed Circuit Board (PCB) of the low power board. The connections of this bus are also available on a three-pin header to allow more sensors to be connected to the system.

µSD: The MCU has a SDIO interface which is used to connect a microSD card. On this card, sensed data can be stored as well as configuration options for the system.

Radio: Different radios can be mounted on the low power board. Submodules to be placed in the socket at the bottom right in Fig. 19 exist. Variations with a LoRa [12] and a IEEE 802.15.4 transceiver exist. The software running on the MCU is able to detect which radio is connected and initializes the according driver modules.

Charge Controller: The *LT3652HV* charge controller is able to charge different kinds, such as LiFe, LiPo as well as sealed lead acid batteries with a current of up to 2 A. The energy source for charging the battery is in this example a PV panel. Voltages and currents are monitored by the MCU.

Switching Regulator: An efficient regulator *LM43603* converts the input voltage of the battery to 3.3 V for powering the components on the board.

Load: Two outputs exist on the board. They can be configured either to output the battery voltage directly with a MOSFET, or to include an optional switching regulator in the output path. If a regulator with an output voltage of 5 V is chosen, SBCs like a *Raspberry Pi* or *BeagleBone* can be powered without additional external components.

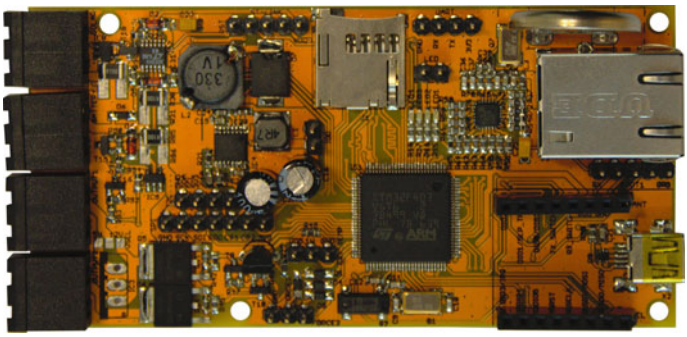


Fig. 19 Photo of Amphisbaena board

6.3 Evaluation

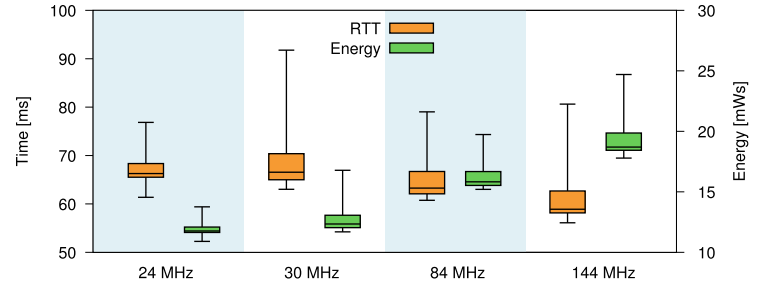
For the nodes handled in this section, the communication performance and the according energy demand are important. Multiple parameters have an influence on the performance of a data transfer. Obvious is the communication link type (Ethernet, WiFi, Bluetooth, IEEE 802.15.4,…), while the software architecture and, of course, the hardware itself also cannot be neglected.

In the aforementioned use case, an Amphisbaena board may communicate using a wireless link either with a SBC, or with another board of the same kind. The variety of communication partners has a great impact on the performance and resulting, the energy demand.

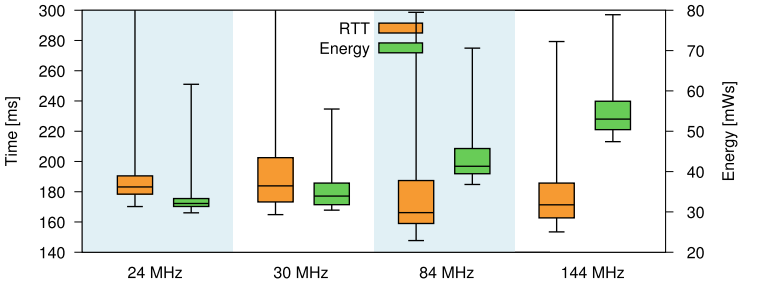
The actual communication protocol to be chosen in the deployment depends on the task of the network as well as its architecture. Often, the traffic pattern occurring in a network can be modeled as a request followed by a response. For control traffic or sensor readings, both packets would be rather small, while cumulated measurements require more data to be transmitted.

For the evaluation, we implemented a simple request/response protocol with payload sizes of 64 and 1024 B. Both, Round Trip Time (RTT) and energy demand of the Amphisbaena board have been captured, from preparing and transmitting the data packet until the answer has been received. GPIO pins have been used to indicate the current state of the transmission. Data is transferred using IEEE 802.15.4 and Ethernet. The latter can be used when acting as a gateway or GSN, even though most nodes deployed on a field will not make use of it. The hardware components of the Ethernet part will be activated if a link is present automatically, which means that no energy will be wasted at nodes not using this interface.

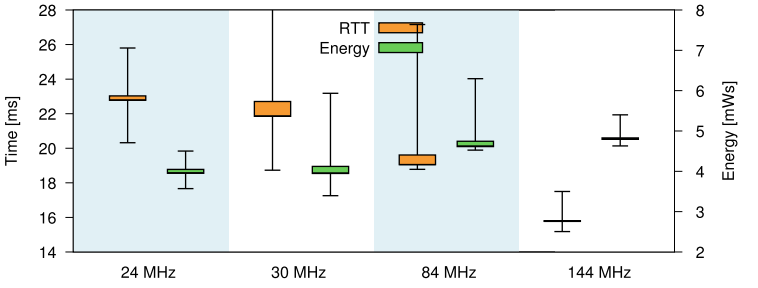
The Amphisbaena board running miniDTN [57] to be evaluated has been configured with different CPU clock frequencies of 24, 30, 84 and 144 MHz. Besides the RTT, the energy consumption has been measured as well, which allows to chose the optimal frequency. The results are shown in Fig. 20. The caption “64/1024 Byte over IEEE 802.15.4 to Raspberry Pi” means that a request with a size of 64 B has



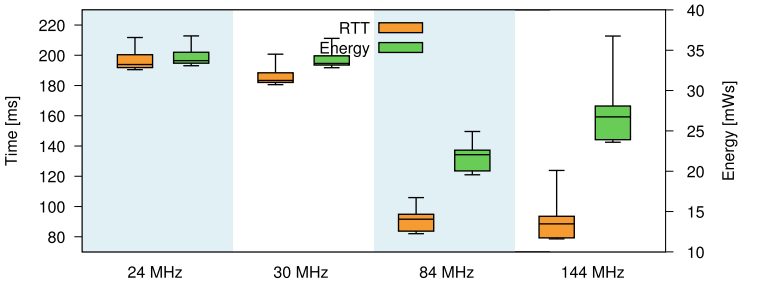
(a) 64/64 Byte over IEEE 802.15.4 to Raspberry Pi



(b) 64/1024 Byte over IEEE 802.15.4 to Raspberry Pi

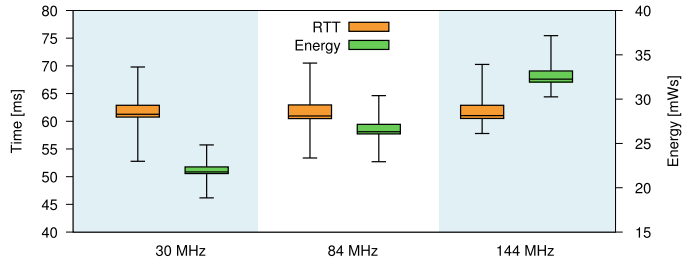


(c) 64/64 Byte over IEEE 802.15.4 to Amphisbaena

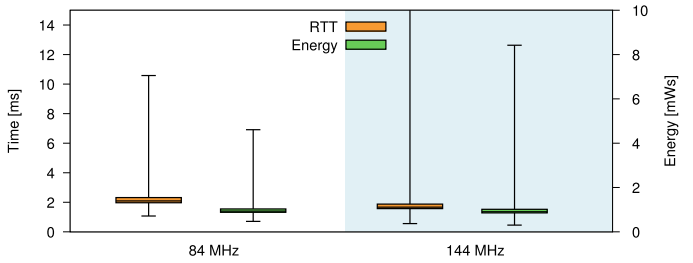


(d) 64/1024 Byte over IEEE 802.15.4 to Amphisbaena

Fig. 20 Data transfers to raspberry Pi and Amphisbaena using IEEE 802.15.4



(a) 64/1024 Byte over Ethernet to Raspberry Pi



(b) 64/1024 Byte over Ethernet to Core i7

Fig. 21 Data transfers to raspberry Pi and Intel Core i7 using Ethernet

been sent and a response of 1024 B has been received, from a Raspberry Pi running IBR-DTN. Orange boxplots show the time from sending the data until the complete reception of the answer in the user space, while the green ones depict the energy demand of the Amphisbaena board which made the request.

The results depicted in Fig. 20 show that the Raspberry Pi running IBR-DTN has a large overhead resulting in a much higher RTT for the measurements. An increased clock frequency of the Amphisbaena board has only a small influence on the RTT while the energy demand increases for higher frequencies. This is a result from the fact that the Amphisbaena has to wait long periods of time for the response.

When communicating with another Amphisbaena board, the lightweight operating system running on both MCUs leads to much lower latencies.

In Fig. 21, we can see again the influence of the partner's communication platform. We are using a wired Ethernet (100 Mbit/s) link. The communication partner is either a Raspberry Pi or a PC based on an Intel Core i7-3770, both running IBR-DTN in the same version. Again, an increased clock frequency does not lead to a lower RTT when exchanging data with a Raspberry Pi (Fig. 21a). Figure 21b reveals that a high processing power combined with a fast communication medium results in very low RTTs which reduces the energy demand significantly.

Concluding, it can be said that many parameters should carefully be chosen for the communication. Although Ethernet itself has a high energy demand, it is much faster than IEEE 802.15.4 and thus leads to a reduced energy consumption—as long as it is availabe and the communication partner is able to handle the data fast enough.

7 Summary

WSNs have been a topic in research for many years already and many different sensor node architectures and designs have been developed and built. This chapter showed important design considerations to make when choosing or developing a wireless sensor node for a specific application. Some nodes available on the market are very generic, like the presented and well renowned TMote Sky. They are a good choice for development and education purposes. Although the basic setup is similar for most sensor nodes, for many real-world applications, hardware specifically designed for the respective mission is required. In order to explain the design process for a mission-oriented sensor node, we presented two exemplary missions with diverse requirements. Firstly, the AAL project was introduced and used to elucidate the hardware development of the sensor node INGA. It is highly mission-oriented towards that use case. Secondly, the dual-platform node Amphisbaena was introduced which was designed to optimally suit the requirements posed by smart farming applications. The evaluation of both of those nodes proved them performing very well in their respective use-cases.

This at the same time does not make them inapplicable for other missions. INGA for example is still rather generic and can thus be used for a variety of applications as it is. However, the USB port and the selection of sensors result in a comparatively high price. In order to further reduce cost and size, even more specialized derivatives have been build. This includes for example Inexpensive Node for Bed-Exit-Detection (InBED) as shown in Fig. 22. Its purpose is to detect people getting out of bed when they should not—e.g. after a surgery. It is only equipped with the processing unit, the transceiver unit and an accelerometer, making it much smaller and less expensive than the original INGA.

After all, many choices have to be made to decide for the optimal hardware for a given mission. While there are many generic sensor nodes available on the market that might be able to serve a plethora of tasks, they can never perform as

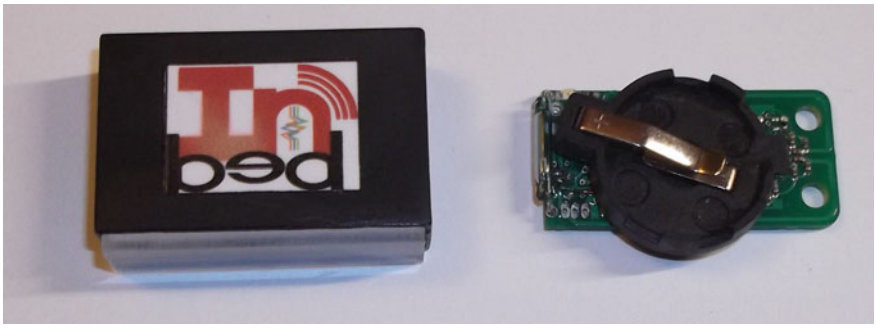


Fig. 22 Specialized InBED sensor node

well in specialized missions, as dedicated hardware specifically designed for that very purpose can. As a result, it is often beneficial to develop new sensor nodes for specific missions instead of using available, more generic ones.

References

1. Doolin, D.M., Sitar, N.: Wireless sensors for wildfire monitoring. Proceedings of SPIE **5765**, 477–484 (2005)
2. Li, Y., Wang, Z., Song, Y.: Wireless sensor network design for wildfire monitoring. In: The Sixth World Congress on Intelligent Control and Automation: WCICA 2006, vol. 1, pp. 109–113 IEEE (2006)
3. Korber, H.-J., Wattar, H., Scholl, G.: Modular wireless real-time sensor/actuator network for factory automation applications. IEEE Trans. Indust. Informat. **3**(2), 111–119 (2007)
4. Frotzscher, A., Wetzker, U., Bauer, M., Rentschler, M., Beyer, M., Elspass, S., Klessig, H.: Requirements and current solutions of wireless communication in industrial automation. In: 2014 IEEE International Conference on Communications Workshops (ICC), pp. 67–72, IEEE (2014)
5. IEEE, IEEE standard for a precision clock synchronization protocol for networked measurement and control systems. IEEE Std 1588–2008 (Revision of IEEE Std 1588–2002), pp. 1–300, July 2008
6. Scheiterer, R.L., Na, C., Obradovic, D., Steindl, G.: Synchronization performance of the precision time protocol in industrial automation networks. IEEE Trans. Instrum. Meas. **58**(6), 1849–1857 (2009)
7. von Zengen, G., Garlichs, K., Schröder, Y., Wolf, L.C.: A sub-microsecond clock synchronization protocol for wireless industrial monitoring and control networks. In: IEEE International Conference on Industrial Technology (ICIT), pp. 1266–1270 IEEE (2017)
8. Akhlaq, M., Sheltami, T.R.: RTSP: an accurate and energy-efficient protocol for clock synchronization in WSNs. IEEE Trans. Instrum. Meas. **62**(3), 578–589 (2013)
9. Dunkels, A., Grönvall, B., Voigt, T.: Contiki—a lightweight and flexible operating system for tiny networked sensors. In: Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I) Nov 2004, Tampa, Florida, USA (2004)
10. Baccelli, E., Hahm, O., Gunes, M., Wahlisch, M., Schmidt, T.C.: RIOT OS: towards an OS for the Internet of Things. In: 2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 79–80 IEEE (2013)
11. Real Time Engineers Ltd.: FreeRTOS—Market leading RTOS (Real Time Operating System) for Embedded Systems with Internet of Things Extensions. http://www.freertos.org/
12. Seller, O., Sornin, N.: Low Power Long Range Transmitter. EP Patent App. EP20, 130, 154, 071, 6 Aug 2014
13. Espressif Systems: ESP32 Datasheet. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (2017)
14. Texas Instruments, Incorporated: A USB-Enabled System-On-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee Applications. http://www.ti.com/lit/gpn/cc2531 (2010)
15. Atmel Corporation: 8-bit AVR Microcontroller with Low Power 2.4GHz Transceiver for ZigBee and IEEE 802.15.4. http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8393-MCU_Wireless-ATmega256RFR2-ATmega128RFR2-ATmega64RFR2_Datasheet.pdf (2014)
16. Moteiv Corporation: Tmote Sky : Datasheet. http://www.bandwavetech.com/download/tmote-sky-datasheet.pdf (2006)
17. Adjih, C., Baccelli, E., Fleury, E., Harter, G., Mitton, N., Noel, T., Pissard-Gibollet, R., Saint-Marcel, F., Schreiner, G., Vandaele, J., et al.: FIT IoT-LAB: a large scale open experimental IoT

testbed. In: IEEE 2nd World Forum on Internet of Things (WF-IoT), vol. 2015, pp. 459–464, IEEE (2015)
18. Lim, R., Ferrari, R., Zimmerling, M., Walser, C., Sommer, P., Beutel, J.: Flocklab: a testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In: Proceedings of the 12th International Conference on Information Processing in Sensor Networks (IPSN) ACM, pp. 153–166 (2013)
19. Doddavenkatappa, M., Chan, M.C., Ananda, A.L.: Indriya: a low-cost, 3d wireless sensor network testbed. In: International Conference on Testbeds and Research Infrastructures, pp. 302–316, Springer (2011)
20. Texas Instruments, Incorporated: MSP430F15x, MSP430F16x, MSP430F161x. Mixed Signal Microcontroller. http://www.ti.com/lit/gpn/msp430f1611 (2011)
21. Texas Instruments, Incorporated: 2.4 GHz IEEE 802.15.4/ZigBee-ready RF Transceiver. http://www.ti.com/lit/gpn/cc2420 (2013)
22. Texas Instruments: MSP430 Flash Device Bootloader (BSL), SLAU319M. http://www.ti.com/lit/ug/slau319m/slau319m.pdf (2017)
23. Alliance, T.: TinyOS 2.1 adding threads and memory protection to TinyOS. In: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, ser. SenSys '08. New York, NY, USA: ACM, pp. 413–414. https://doi.org/10.1145/1460412.1460479 (2008)
24. O'Donovan, T., Brown, J., Büsching, F., Cardoso, A., Cecelio, Do O, J., Furtado, P., Gil, P., Jugel, A., Pöttner, W.-B., Roedig, U., Silva, J., Sreenan, C., Vassiliou, V., Voig, L.W.T., Zinonos, Z.: The GINSENG system for wireless monitoring and control: design and deployment experiences. ACM Trans. Sensor Netw. (TOSN), vol. 10, no. 3, accepted for publication, Aug 2014
25. Durvy, M., Abeillé, J., Wetterwald, P., O'Flynn, C., Leverett, B., Gnoske, E., Vidales, M., Mulligan, G., Tsiftes, N., Finne, N., Dunkels, A.: Making Sensor Networks ipv6 Ready. In: Proceedings of the 6th ACM Conference on Embedded Network Sensor System ser. SenSys '08. New York, NY, USA: ACM, pp. 421–422. https://doi.org/10.1145/1460412.1460483 (2008)
26. Eichelberg, M., Hein, A., Büsching, F., Wolf, L.: The GAL middleware platform for AAL. In: Proceedings of the 12th IEEE International Conference on e-Health Networking Applications and Services (Healthcom), pp. 1–6 (2010)
27. Chen, J., Kwong, K., Chang, D., Luk, J., Bajcsy, R.: Wearable sensors for reliable fall detection. In: proceedings of the 27th Annual International Conference of the Engineering in Medicine and Biology Society. IEEE-EMBS, vol. 2005 (01), pp. 3551–3554 (2005)
28. Marschollek, M., Wolf, K.H., Gietzelt, M., Nemitz, G., Meyer zu Schwabedissen, Haux, R.: Assessing elderly persons' fall risk using spectral analysis on accelerometric data—a clinical evaluation study," in *the 30th Annual International Conference of the Engineering in Medicine and Biology Society, IEEE-EMBS 2008*, 2008, pp. 3682–3685
29. Greene, B., McGrath, D., O'Donovan, K., O'Neill, R., Burns, A., Caulfield, B.: Adaptive estimation of temporal gait parameters using body-worn gyroscopes. In: 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), vol. 4, pp. 1296–1299 31, Sept 2010
30. Bianchi, F., Redmond, S., Narayanan, M., Cerutti, S., Celler, B., Lovell, N.: Falls event detection using triaxial accelerometry and barometric pressure measurement. In: Proceedings of the 31th Annual International Conference of the Engineering in Medicine and Biology Society, IEEE-EMBC 2009, pp. 6111–6114 Sept 2009
31. Polastre, J., Szewczyk, R., Culler, D.: Telos: enabling ultra-low power wireless research. In: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, IPSN 2005. Piscataway, NJ, USA: IEEE Press, 2005, p. 48
32. Lorincz, K., Chen, B.-R., Challen, G.W., Chowdhury, A.R., Patel, S., Bonato, P., Welsh, M.: Mercury: a wearable sensor network platform for high-fidelity motion analysis. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, ser. SenSys '09. New York, NY, USA: ACM, pp. 183–196. https://doi.org/10.1145/1644038.1644057 (2009)
33. Welk, G.J., McClain, J.J., Eisenmann, J.C., Wickel, E.E.: Field validation of the MTI actigraph and bodyMedia Armband monitor using the IDEEA monitor. Obesity **15**(4), 918–928. https://doi.org/10.1038/oby.2007.624 (2007)

34. Büsching, F., Kulau, U., Wolf, L.: Architecture and evaluation of INGA an inexpensive node for general applications. In: 2012 IEEE Sensors, Oct 2012, pp. 1–4
35. Atmel Corporation: AVR2016: RZRAVEN Hardware User's Guide, 2011, Rev. 8117D-AVR-04/08. http://www.atmel.com/dyn/resources/prod_documents/doc8117.pdf
36. Büsching, F., Figur, A., Schürmann, D., Wolf, L.: Poster: utilizing hardware AES encryption for WSNs. In: Proceedings of the 10th European Conference on Wireless Sensor Networks, ser. EWSN 2013, Ghent, Belgium, Feb 2013, pp. 33–36. http://www.ibr.cs.tu-bs.de/papers/buesching-ewsn2013.pdf
37. von Zengen, G., Schröder, Y., Rottmann, S., Büsching, F., Wolf, L.C.: No-cost distance estimation using standard WSN radios. In: Proceedings of the 35th Annual IEEE International Conference on Computer Communications (INFOCOM 2016), San Francisco, USA, Apr 2016
38. Atmel Corporation: AVR2006: Design and characterization of the Radio Controller Board's 2.4GHz PCB Antenna, Rev. 8095A-AVR-08/07. http://www.atmel.com/dyn/resources/prod_documents/doc8095.pdf (2017)
39. Büsching, F., Kulau, U., Gietzelt, M., Wolf, L.: Comparison and validation of capacitive accelerometers for health care applications. Comput. Methods Prog. Biomed. **10**
40. Future Technology Devices International Ltd.: TBit Bang Modes For The FT232R and FT245R, Application Note AN 232R-01 (2012)
41. Dunkels, A.: Rime—a lightweight layered communication stack for sensor networks. In: Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session, Delft, The Netherlands. Citeseer (2007)
42. Kulau, U., Büsching, F., Wolf, L.: A node's life: increasing WSN lifetime by dynamic voltage scaling. In: Proceedings of the 9th IEEE International Conference on Distributed Computing in Sensor Systems 2013 (IEEE DCoSS 2013), Cambridge, USA, May 2013
43. Österlind, F., Dunkels, A., Eriksson, J., Finne, N., Voigt, T.: Cross-level sensor network simulation with COOJA. In: Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006), Tampa, Florida, USA, Nov 2006. http://www.sics.se/nes/osterlind06crosslevel.pdf
44. Titzer, B.L., Lee, D.K., Palsberg, J.: Avrora: scalable sensor network simulation with precise timing. In: Proccedings of the 4th International Symposium on Information Processing in Sensor Networks, IPSN. IEEE, vol. 2005, pp. 477–482 (2005)
45. Kulau, U., Rottmann, S., Schildt, S., van Balen, J., Wolf, L.C.: Undervolting in real world WSN applications: a long-term study. In: Proceedings of the International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 9–16, IEEE (2016)
46. Langendoen, K., Baggio, A., Visser, O.: Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. In: Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium, ser. IPDPS vol. 2006, 8 (2006)
47. Kaewmard, N., Saiyod, S.: Sensor data collection and irrigation control on vegetable crop using smart phone and wireless sensor networks for smart farm. In: IEEE Conference on Wireless Sensors (ICWiSE), pp. 106–112, IEEE (2014)
48. Ojha, T., Misra, S., Raghuwanshi, N.S.: Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges. Comput. Electron. Agr. **118**, 66–84 (2015)
49. Priya, S., Inman, D.J.: Energy Harvesting Technologies, vol. 21. Springer (2009)
50. Beutel, J., Buchli, B., Ferrari, F., Keller, M., Zimmerling, M., Thiele, L.: X-Sense: sensing in extreme environments. In: Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE 2011, pp. 1–6 (2011)
51. Navarro, M., Davis, T.W., Liang, Y., Liang, X.: A study of long-term WSN deployment for environmental monitoring. In: Proceedings of the 24th IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), Sept 2013
52. Wennerstrom, H., Hermans, F., Rensfelt, O., Rohner, C., Norden, L.-A.: A long-term study of correlations between meteorological conditions and 802.15. 4 link performance. In: 2013 10th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), pp. 221–229. IEEE (2013)

53. Boano, C.A., Wennerström, H.M., Zúñiga, A., Brown, J., Keppitiyagama, C., Oppermann, F.J., Roedig, U., Nordén, L.-Å., Voigt, T., Römer, K.: Hot packets: a systematic evaluation of the effect of temperature on low power wireless transceivers. In: Proceedings of the 5th Extreme Conference on Communication (ExtremeCom), pp. 7–12. ACM, Aug 2013
54. Boano, C.A., Zuniga, M.A., Brown, J., Roedig, U., Keppitiyagama, C., Roemer, K.: TempLab: a testbed infrastructure to study the impact of temperature on wireless sensor networks. In: Proceedings of the 13th International Conference on Information Processing in Sensor Networks (IPSN '14), pp. 95–106. ACM, Apr 2014
55. Schmidt, F., Ceriotti, M., Hauser, N., Wehrle, K.: If you can't take the heat: temperature effects on low-power wireless networks and how to mitigate them. In: 12th European Conference on Wireless Sensor Networks (EWSN 2015). Feb 2015
56. Jackson, R.D., Idso, S., Reginato, R., Pinter, P.: Canopy temperature as a crop water stress indicator. Water Resour. Res. **17**(4), 1133–1138 (1981)
57. Rottmann, S., Hartung, R., Käberich, J., Wolf, L.C.: Amphisbaena: a two-platform DTN node. In: The 13th International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2016), Brasilia, Brazil, Oct 2016
58. Gernert, B., Rottmann, S., Wolf, L.C.: Poster: PotatoMesh–A Solar Powered WSN Testbed. Paderborn, Germany (2016)

Failure Handling in RPL Implementations: An Experimental Qualitative Study

Agnieszka Paszkowska and Konrad Iwanicki

Abstract The IPv6 Routing Protocol for Low-power and Lossy Networks (RPL) is a recognized standard for routing packets in low-power wireless networks. Its two popular implementations—TinyRPL and ContikiRPL—have been used for both research and commercial purposes. However, despite their wide adoption, qualitative studies of their behavior under various types of failures are essentially lacking. Therefore, in this chapter, we aim to bridge this gap in a manner that may be of interest to both researchers and practitioners. More specifically, we evaluate the two implementations of RPL in a range of link and node failure scenarios. We show that whereas the implementations handle well some classes of failures, for others they exhibit undesirable behaviors or even fail completely. The results thus identify failure scenarios handling which may require additional attention before employing the implementations in real-world dependable embedded systems.

1 Introduction

A routing protocol is practically indispensable in embedded systems involving wireless low-power and lossy networks (LLNs). It allows network nodes that are out of each other's radio range to communicate by forwarding data packets via other, intermediate nodes. In effect, it alleviates several real-world problems associated with implementing and deploying LLN-based embedded systems, notably communication obstacles in the target environment or large dimensions of the environment

A. Paszkowska (✉) · K. Iwanicki
Faculty of Mathematics, Informatics and Mechanics,
University of Warsaw, ul. Banacha 2, 02-097 Warszawa, Poland
e-mail: ap321142@students.mimuw.edu.pl

K. Iwanicki
e-mail: iwanicki@mimuw.edu.pl

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_3

itself. If, in addition, the protocol is self-organizing, it can also significantly reduce the administrative costs inherent in configuring and maintaining such systems.

The IPv6 Routing Protocol for Low-power and Lossy Networks (RPL) is meant as such a protocol [36]. RPL is IETF's standard, incorporating state-of-the-art solutions, developed specifically to address the peculiar requirements of LLNs. It has both proprietary and open-source implementations, among which TinyRPL for TinyOS [26] and ContikiRPL for ContikiOS [6] are arguably the best known ones [23]. In particular, TinyRPL and ContikiRPL have become inherent components of LLN protocol stacks in both research-oriented deployments and commercial solutions. In short, to deliver their functionality, multiple embedded systems rely on RPL-based networking provided by these implementations.

Nevertheless, despite the growing adoption of RPL's implementations, many of their aspects remain largely unexplored. One of such aspects is the behavior of the implementations in various failure scenarios. As we elaborate in subsequent sections, whereas the performance of RPL's implementations has been studied quantitatively, qualitative studies of their behavior under specific types of failures are essentially lacking. Without such studies, however, it is not clear to what extent the implementations can be relied upon in dependable embedded systems.

For this reason, here we aim to bridge this gap. To this end, we evaluate the two aforementioned state-of-the-art implementations of RPL—TinyRPL and ContikiRPL—in a range of failure scenarios. As the evaluation environment, we employ low-level simulators: TOSSIM [25] and COOJA [31], respectively. The scenarios cover in turn crashes of specific nodes and links, so as to affect the network topology in a particular manner, which would be hard to achieve with experiments studying robustness purely statistically. We show that whereas some of the considered failures are handled well by the implementations, for others the implementations exhibit undesirable behaviors or even fail completely. We also identify causes of such behaviors, which may be of particular interest to practitioners who are planning to use the implementations in their embedded systems. All in all, our results shed new light on the dependability of RPL's popular implementations.

The rest of this chapter is organized as follows. Section 2 gives an overview of RPL. Section 3 surveys related work. Section 4 presents our experimental methodology. Sections 5–8 discuss the results for the subsequent failure scenarios. Section 9 concludes and outlines possible future work.

2 Overview of RPL

To study the behavior of RPL's implementations, let us first give an overview of RPL itself, including details on how it provisions routes, what data structures it uses to maintain the routes and how it handles failures and topology changes, to name just a few examples. In addition to discussing the standardized aspects of the protocol, we highlight unspecified issues, which are left open to implementations.

Although many routing techniques exist for LLNs, including shortest path routing [9], compact routing [28], hierarchical routing [19], and routing by means of geographic [22] or virtual [8] coordinates, to name a few representative examples, RPL is in principle a centralized protocol. Despite a potentially inferior performance of this technique [20], its choice can likely be attributed to simplicity.

Being a centralized protocol, RPL combines solutions for two basic traffic patterns: so-called *upward routing*, in which all low-power nodes forward their packets to a common (central) destination node, typically a border router, with so-called *downward routing*, from the border router to the nodes. These two basic patterns enable more complex ones, in particular, any-to-any routing between arbitrary nodes. However, it is upward routing, implemented by means of a distance-vector algorithm, that is the foundation of the protocol: If it does not work correctly, the other patterns will not work either. Consequently, for brevity, in our studies we focus solely on upward routing.

2.1 Preliminary Information

Recall that a routing protocol is responsible for directing packets in a LLN. More specifically, a node receiving a packet from a given *source node* to a given *destination node* needs to select the *next-hop node*, to which the packet will be forwarded, so that it can finally reach its destination. In LLNs, the next hop for a node is chosen from the nodes within the node's radio range. In RPL, such nodes are called the node's *neighbors*.

However, not every neighbor is a good candidate for the node's next hop. This is because transmitting a packet to a next hop should bring the packet closer to the destination, or, stated more formally, it should decrease a cost of the packet reaching the destination.

Therefore, each node in RPL's upward routing distinguishes a subset of neighbors that have a lower cost of sending a packet to the destination than this node. Such neighbors are called the node's *parents*. The node uses such a parent set to select a single *preferred parent*: the primary parent chosen as a default next hop for packets forwarded by the node to the given destination. The cost of sending a packet from a node to the destination is in turn called the node's *rank*.

Since when selecting a preferred parent, the node should in principle consider the neighbors with the lowest ranks, the rank aims to reflect the node's distance from the destination. However, RPL's specification does not state exactly how to measure this distance. In particular, the node's rank can depend on the requirements of a specific application and the choice of a metric to be optimized globally in the network when routing packets [35], for example, hop count, latency, or estimated transmission count, ETX [5].

Conceptually, nodes and the wireless links to their parents form a directed graph. Since when the network is stable a node's rank is always greater than its parents' ranks, the graph does not contain cycles and, consequently, is a directed acyclic graph

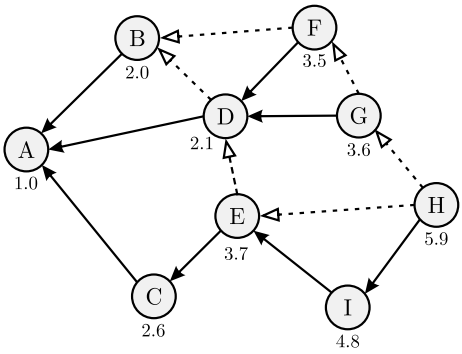


Fig. 1 An example of a DODAG

(DAG). The destination, for example, a border router, is the sink in the DAG: it has no outgoing edges. Therefore, the graph is often referred to as a destination-oriented DAG, abbreviated as *DODAG*. The destination—the sink of the graph—is in turn called the *root* of the DODAG.

Figure 1 presents a sample DODAG rooted at node A, with nodes' ranks displayed as numbers. Links between nodes and their parents are represented by lines with arrows pointing at the parent. A solid line leads to a preferred parent; a dashed one to an alternative parent. Node A's rank equals 1.0 and is the smallest rank in the network because A is the DODAG's root. For the same reason, A does not have any outgoing edges. D, an internal node, has in turn five neighbors: A, B, E, F, and G. Two of them, A and B, have their ranks lower than D's rank, 2.1, and, consequently, D considers them as parents: A is D's preferred parent and B is an alternative one. Although D has chosen as its preferred parent the neighbor with the smallest rank among its parents, such a choice is not imposed by the protocol. For example, E has C with rank 2.6 as its preferred parent, despite having D with a lower rank, 2.1, in its parent set. The rest of D's neighbors, E, F, and G, have their ranks greater than D's rank, and, therefore, are D's children in the DODAG. Moreover, D is both F's and G's preferred parent. As a result, when, for instance, G wants to send a packet to the root node A, it first transmits the packet to D, which forwards the packet directly to A, yielding a hop count of 2. In contrast, the maximum number of hops to deliver a packet in the DODAG is 4 and is required when the packet originates at node H.

2.2 Packet Forwarding

RPL's DODAG thus describes all routes via which nodes can forward packets to the root node. However, the actual packet forwarding at each node is delegated by RPL to the node's IPv6 stack. To this end, RPL running at the node registers the node's preferred parent's IPv6 address as the default route in the node's IPv6 routing table. Based on this table, the node's IPv6 stack selects next hops for forwarded packets.

Nevertheless, in spite of the delegated packet forwarding, RPL's implementations should provide endpoints for inserting and verifying a special option in the headers of routable packets [15]. The option contains, among others, information used for loop and error detection: a forwarding node's rank and three flags: 'O', 'R', and 'F'.

Recall that a node's rank is meant to be greater than its preferred parent's rank, which aims to avoid routing loops during normal operation. However, such loops may appear under failures, notably when the node has outdated information on its neighbors' ranks. Therefore, an additional loop detection mechanism is needed to quickly react to such rank inconsistencies. The mechanism detects an inconsistency whenever the direction of a forwarded packet (the 'O' flag in the option) does not match the rank relationship between the node transmitting the packet and the neighbor receiving it. One case is when the packet is going upward (the 'O' flag is clear) but the transmitter's rank in the packet is lower than the receiver's rank. Another case is the packet going downward (the 'O' flag is set) with the transmitter's rank in the packet being greater than the receiver's.

2.3 Parent Selection and Rank Computation

To register a default route, each node needs to select its preferred parent. This process is inherently coupled with the node's rank computation: The node's rank and preferred parent are chosen with a so-called *objective function*, which we abbreviate as *OF*. As far as objective functions are concerned, RPL's specification allows for much flexibility. First, new objective functions can be introduced, so that computed ranks and the method for selecting preferred parents would meet the requirements of particular applications. Second, OFs are configurable per node.

To illustrate how RPL uses objective functions, recall that each node maintains a parent set, a subset of neighbors from which the node chooses its preferred parent. An entry in the parent set contains, among others, a parent's address, rank and routing metric values for the parent and/or the link from the node to the parent. Since the node computes its own rank locally, it has to exchange information on its ranks with its neighbors by means of control traffic. The entries in the node's parent set are thus inserted and updated as a result of receiving control messages, which will be discussed in more detail shortly.

The parent set is an input to an objective function, which the node uses to choose its preferred parent. Parent selection should in principle be performed whenever the parent set has potentially changed, for example, upon reception of a control message with a neighbor's rank update or parent unreachability detection [30]. As soon as the node selects its preferred parent, it registers the route to the parent in its IPv6 routing table as the default route.

The objective function is also responsible for computing the node's rank after a change of the preferred parent or the metrics associated with the parent. The algorithm for rank computation is OF-dependent. Nevertheless, RPL's specification provides guidelines to be followed by every OF.

First of all, the rank value needs to reflect the node's distance from the root and has to be strictly increasing along the routes in the DODAG. To ensure the latter, the protocol uses a *MinHopRankIncrease* parameter, which defines the smallest permitted increase in rank per hop. As a result, the root's rank must be the smallest rank in the DODAG and every other node's rank must be higher than any of its parents' ranks and exceed its preferred parent's rank by at least *MinHopRankIncrease*. This feature is made use of by the aforementioned loop detection mechanism.

Furthermore, to enable breaking loops in the DODAG, which may occur upon failures, a node's rank must not increase by more than *MaxRankIncrease* within one so-called *version* of the DODAG. Namely, a node keeps track of the smallest rank it has computed in the current DODAG version, and, whenever its rank starts to exceed that smallest rank by more than *MaxRankIncrease*, it concludes that the DODAG is broken and disconnects from it by adopting a null preferred parent and an infinite rank. However, since the change in its rank may be permanent, for instance, as a result of a massive failure, DODAG versioning enables a complete reconstruction of the DODAG. Such as process is initialized by the root generating the DODAG's new version number, either periodically or as a result of an external event (e.g., a request from the network administrator). Upon discovering the new version, a node can choose a completely new rank, not constrained by its rank in the old version.

Although parameters *MinHopRankIncrease* and *MaxRankIncrease* exist irrespective of an objective function, their values may potentially depend on the particular OF employed. Currently, there are two popular OFs.

First, all implementations of RPL are required to implement the Objective Function Zero, OF0 [33]. In OF0, a node's rank equals the count of hops from the root to the node. The criteria taken into account in the parent selection process include both the candidate's rank and the quality of the link to the candidate. It is, however, the candidate's rank that influences the result the most.

Second, in practice the Minimum Rank with Hysteresis Objective Function, MRHOF [12] is commonly used. With MRHOF, a node uses a metric value retrieved from an additional option in control messages to evaluate parents and compute its rank. Various metrics can be used, for example, hop count, ETX, latency, though they need to be additive. The node's rank is calculated from the candidate parent's metric value incremented by the link metric to the candidate parent. The candidate with the lowest path cost is chosen as the node's preferred parent but only if it is significantly better than the current parent. This mechanism of not choosing the best parent unless truly beneficial is called *hysteresis* and is introduced in MRHOF to make the DODAG more stable.

2.4 Control Traffic

The algorithm for choosing parents and calculating ranks generates control traffic through which the nodes exchange information required by OFs. RPL introduces several types of control messages, implemented as ICMPv6 [4] messages.

The first one, DODAG Information Object (DIO) messages, is responsible for advertising nodes' ranks: A DIO message sent by a node is an advertisement that the node can serve for its neighbors as a next hop on their routes to the DODAG root. The root of a DODAG systematically sends such a message. A neighbor receiving the message can join the DODAG and select the sender node as its preferred parent. When it sends its own DIO, its neighbors can do the same, and so on.

A DIO message thus needs to contain all the information necessary for joining a DODAG, such as the DODAG's identifier, version number and configuration, and for considering the sender as a parent, for instance, information on the sender's rank and metric values. The configuration of a DODAG contains, among others, the identifier of the objective function to be used by the nodes to choose their parents and compute their ranks and the two aforementioned parameters used in rank computation and maintenance: *MinHopRankIncrease* and *MaxRankIncrease*.

There is a trade-off between keeping the time of reaction to network changes low and generating little control traffic for DODAG maintenance. To exploit this trade-off, each node employs a so-called Trickle timer [27] for coordinating the control traffic. After starting, the node sets its timer period to t_{min} (on the order of milliseconds) and doubles the next period whenever the previous one finishes until the period reaches t_{max} (on the order of minutes or even hours). However, the node can reset the period back to t_{min} whenever it observes an important change or an inconsistency in the DODAG.

The Trickle timer serves as the node's scheduler for DIO messages. At the beginning of each period, the node schedules the timer to a random time in the second half of the period. When the timer fires, the node broadcasts a DIO message to its neighbors unless the number of DIO messages it received in the current period exceeds some threshold. As a result, initially, nodes exchange many DIO messages and thus quickly construct a DODAG. However, after this short period, the frequency of DIO messages gradually decreases and stays low as long as the network is stable. In effect, not only can changes in the network be accounted for rapidly but also few messages are exchanged when the DODAG does not change, which minimizes the control traffic.

The second type of RPL control messages, DIS (DODAG Information Solicitation) messages, are used by nodes to actively ask their neighbors to provide information on the DODAGs they are aware of. This results in a quicker reaction to network changes at the cost of an increase in control traffic. A node broadcasts such DIS messages periodically until it joins a DODAG. In contrast, when the node receives such a message, it resets its Trickle timer so that it can quickly send back a DIO response.

Another use case for DIS messages is probing. The aim of probing is to keep parent routing metrics up-to-date. Since probes are unicast messages, the response DIOs are sent directly to the probe's sender without a reset of the responding node's Trickle timer.

The other control messages introduced by RPL are not related to upward routing. Therefore, we omit them here for brevity.

2.5 Open Issues

RPL's specification does not provide solutions for all issues regarding the protocol, leaving them open to implementations. To give some example, while it is specified when the Trickle timer for DIO messages must be reset, the list is not exhaustive. Implementations may thus choose to reset the timer in other situations. Another example is the parent choice and rank computation, which are both OF- and implementation-dependent. Nevertheless, the specification provides guidelines that must be followed by all OF implementations. Finally, issues regarding routing metric maintenance and neighbor unreachability detection are unspecified and left open to implementations. All in all, the implementations have some freedom with respect to the way they provide RPL's functionality. This allows for adapting them to various applications, which is another advantage in embedded systems.

2.6 RPL Implementations

As a result of the standardization, RPL has been implemented in several operating systems for low-power wireless embedded devices. There thus exist both proprietary and open-source implementations of RPL, among which TinyRPL for TinyOS [26] and ContikiRPL for ContikiOS [6] are arguably the most widely known ones [23]. Therefore, we will focus on these two implementations in our analysis.

In short, both implementations are built on top of the IPv6 stacks of their underlying operating systems. They provide all basic features described in RPL's specification. In addition, supported by the research community, they try to integrate state-of-the-art solutions for the issues the specification leaves open. In particular, from the perspective of our study, it is worth to mention that for routing metric maintenance and neighbor unreachability detection, the implementations employ algorithms drawing from numerous studies on the performance of low-power wireless communication and wireless link quality estimation.

What is more, the implementations have been widely tested and deployed in the real world. In effect, various embedded systems have appeared that rely on these implementations to deliver their own functionality.

3 Related Work

Because of this increasing popularity, RPL has been extensively evaluated in both simulations and the real world. However, whereas RPL's implementations have been studied thoroughly from the performance perspective, their behavior under specific types of failures has received significantly less research attention.

To start with, initial empirical studies of RPL focused on its efficiency. Results obtained in early experiments, both in simulations [3, 34] and on testbeds [3, 10], were promising but also unveiled first problems resulting from RPL's underspecification [3]. In particular, it was shown that although RPL successfully detected simple failures [10], its DODAG reconstruction algorithms were not very efficient [34].

As RPL's specification was being clarified and improved, the protocol's performance was being further evaluated in various settings. To illustrate, Gaddour et al. [11] studied the efficiency of ContikiRPL during DODAG formation for different network topologies. Istomin et al. [17], in turn, evaluated the performance of actuation in downward routing. Likewise, RPL's performance was analyzed for different parameter configurations, including objective functions [2] and network-layer metrics [16], to name just two examples. Finally, studies on interoperability of the two aforementioned implementations of RPL—ContikiRPL and TinyRPL—showed that while both implementations perform well on their own, they demonstrate surprising performance artifacts when run together [23].

Furthermore, RPL's performance has been compared to that of other routing protocols, for instance, the Collection Tree Protocol [10], dissemination-based protocols [17], and protocols for Internet-enabled wireless sensor networks [32]. Recently, RPL has also been extended into new routing protocols or even entire protocol suites [1, 7, 29].

All in all, those research activities have made RPL an attractive solution for industry, where dependability is an important feature. As a result, RPL's implementations were evaluated in harsh conditions: under different levels of radio interference [13, 29] and in various failure scenarios, featuring local failures [24], failures of large groups of randomly selected nodes [14], of a few albeit critical nodes [14, 21], and of the DODAG root [18]. Although earlier of those results [14, 21, 24] suggested that RPL's implementations are by and large stable and correctly handle failures, our recent study [18] showed that those conclusions may have been too optimistic. More specifically, we observed that ContikiRPL has surprising problems dealing with a crash of the DODAG root, which is problematic in real-world systems because, being typically more complex than sensor nodes and relying on a tethered power supply, DODAG roots are prone to failures such as power outages.

Therefore, as a follow up on this observation, here we evaluate two popular implementations of RPL—TinyRPL and ContikiRPL—in a much broader range of failure scenarios and with different configuration parameters. Such a qualitative study may thus be of interest to both researchers and practitioners because it allows for a better understanding of conditions under which the implementations may be relied upon.

4 Experimental Methodology

Let us start our study by describing the experimental setup we used for the two implementations of RPL. We discuss the environments in which the implementations were evaluated, including the experimental application, configuration parameters,

and performance metrics. We also explain the link and node failure scenarios in which we tested the implementations.

4.1 Experimental Environments

To facilitate reproduction of our results, we evaluated the implementations in publicly available simulators. For the TinyOS implementation of RPL, TinyRPL, we employed the TinyOS simulator, TOSSIM [25]. Similarly, for the ContikiOS implementation of RPL, ContikiRPL, we utilized the Contiki network simulator, COOJA [31]. Both environments are low-level simulators: They normally simulate actual implementations of protocols—not their simplified models—and the obtained results typically well predict the protocol's real-world behavior.

As to the implementations themselves, we analyze the latest version of TinyRPL, that is, one from January 5, 2017. In contrast, when it comes to ContikiRPL, we focus mainly on the last stable version, 3.0, available at the time of writing this chapter rather than on the latest one from the repository, that is, the one from January 5, 2017. This is because the development version performs worse compared to the stable one, which we highlight in our experiments.

4.2 Experimental Settings

The experimental application, which generated network traffic to be routed by RPL, was designed to model a common communication pattern in LLNs: all-to-one data collection. More specifically, each node generated short, one-frame data packets that were forwarded by the network to the root. The interval between two consecutive packets generated by a node was chosen at random between T and $2T$ time units, where T was a configuration parameter. This resulted in relatively uniform multipoint-to-point traffic.

The experimental runs of the application presented here did not use any radio duty cycling techniques, but we did verify that the results with duty cycling did not diverge from the presented ones. This choice is because duty cycling is a task of the link layer, not RPL, and, as such, should not influence the *correctness* of RPL's implementations. Unless the network operates close to its maximal capacity, which is usually not a normal situation in LLNs, and hence is not the case in our experiments, the only observable effects on RPL of employing duty cycling are higher latencies and lower throughput.

In order to test the implementations under various conditions, we conducted experiments with different configuration parameters. Nevertheless, we provide default values for the parameters that, unless noted otherwise, were used in the experiments. The default duration of an experiment was 1 simulated hour. Nodes generated packets to be forwarded to the root every 10–20 s ($T = 10$ s). The distinguished node

acting as the DODAG root was the node with id 0. The highest possible increase in a node's rank within a DODAG version (*MaxRankIncrease*) was by default 7, while the maximum number of hop-by-hop retransmissions of a packet was 5 per hop. Finally, the experiments used either OF0 or MRHOF as the objective function.

We tested the implementations in so-called *unit-disk* topologies, which are often used in theoretical analyses. In our case, they consisted of 121 nodes evenly distributed over an 11-by-11 grid. The radio range of each node was a circle with the center at the node and radius r, where r ranged from 1 to 4. In other words, each node had a perfect link to every node at a distance lower than or equal to r and did not have a link to any node at a distance greater than r. While this model is an idealized one, it is suitable for our purposes: If a protocol does not behave correctly in this idealized model, it is highly unlikely that it will behave correctly in the real world. Intuitively, in the unit-disk model, a node can detect with a perfect accuracy whether a link is up or down, which is crucial for proper neighbor unreachability detection. In contrast, in the real world, there is no perfect failure detector. In particular, there can be false positives that trigger unnecessary topology changes.

The results presented in the remainder of this chapter were gathered during representative runs of the test application. Their analysis focuses mainly on the correctness but also on the efficiency of the evaluated implementations. The first group of metrics aims to examine the DODAG construction process. To this end, we tracked the number of nodes with a preferred parent and a valid path to the root with respect to time, and an average rank of a node in the network. Second, we evaluated the stability of the constructed DODAG by tracking the number of nodes' preferred parent changes, Trickle timer resets, and generated control messages. Finally, we used the last group of metrics, including metrics related to the network traffic and end-to-end delivery rates, to evaluate the efficiency of resource usage by the nodes in the network and the reliability of the network.

4.3 Experimental Scenarios

The experimental scenarios were designed to examine how the considered implementations of RPL behave in various conditions, in particular, under different link and node failures.

However, the first scenario was free of failures and aimed to analyze how long it takes the implementations to build the DODAG and how the implementations behave when the network is stable. These results are used as a baseline for analyzing the results of subsequent experiments.

The second group was experiments with link failures. The scenarios in this group included failures of both individual links and sets of links but here we focus on single-link failures. None of the failures resulted in a network partition. Moreover, we analyzed the consequences of a link failure depending on the importance of the link. In particular, we considered experimental scenarios in which a failing link was

responsible for forwarding packets from either a large part of the network or just a single node.

The next group of experimental scenarios concerned failures of non-root nodes. Similarly to the previous group, the scenarios included failures of both individual nodes and sets of nodes. Like previously, none of the failures analyzed in this group resulted in a network partition. The failing nodes were either important ones, responsible for forwarding packets from a large part of the network, or leaf nodes, not forwarding any packets except for their own. For brevity, however, we focus on failures of only important nodes.

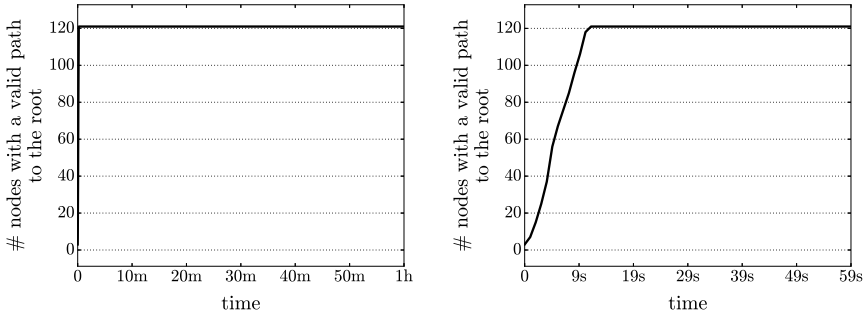
Finally, the last group encompassed scenarios with failures leading to network partitions. One example was failures of DODAG roots. Other examples are both link and node failures as a result of which all paths from some nodes to the root were broken. Nevertheless, we focus on the DODAG root failures as they are extreme cases of network partitions. Additionally, we tested scenarios in which the failing links or nodes recovered after failures, thus rebuilding the broken paths.

All in all, our experimental scenarios cover a broad spectrum of crash failures that may occur in the real world. As such, they truly stress the considered implementations of RPL, so that we can study various claims about their fault tolerance.

5 Experiments Without Failures

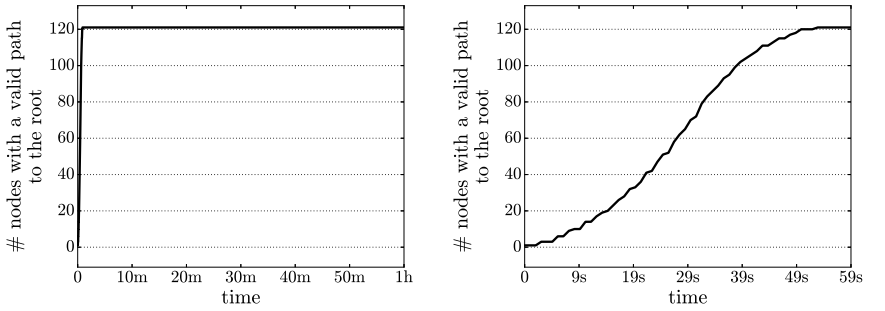
As mentioned in the previous section, we begin our analysis by presenting the results of experiments without failures. Recall that the evaluation of the implementations in the failure-free environment aims to examine how quickly the implementations construct DODAGs and whether the DODAGs reach stable states.

To start with, Fig. 2 presents the number of nodes with a valid path to the DODAG root during the course of a representative experiment conducted with TinyRPL. We define a valid path from a node to the root as a sequence of nodes starting at the node



(a) unit disk, $r = 1$; duration = 1h; MRHOF **(b)** unit disk, $r = 1$; duration = 1m; MRHOF

Fig. 2 Duration of DODAG construction in TinyRPL



(a) unit disk, $r = 1$; duration = 1h; MRHOF **(b)** unit disk, $r = 1$; duration = 1m; MRHOF

Fig. 3 Duration of DODAG construction in ContikiRPL

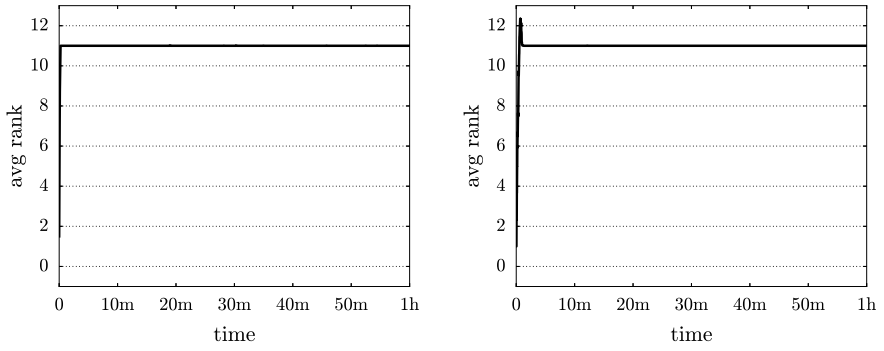
and ending at the root in which (1) all nodes are correct and (2) every two consecutive nodes are connected by a correct link and are in a child–preferred parent relation. We consider a DODAG as constructed when each node has a valid path to the DODAG root and is thus able to forward packets to the root.

It can be observed in the figure that TinyRPL constructs a DODAG within seconds. More specifically, as shown in Fig. 2b, plotting the metric values in the first minute of the experiment, the entire construction process took only 13 s for the most sparse of the analyzed topologies: the unit-disk topology with radius 1. Moreover, as can be observed in Fig. 2a, once constructed, the DODAG was stable during the remaining period of the experiment, that is, all nodes always had valid paths to the root.

Figure 3 presents the same plots for ContikiRPL. It can be observed in Fig. 3b that constructing the DODAG for the same topology (i.e., the sparsest one) took ContikiRPL almost a minute, which is 4 times more than in the case of TinyRPL. The reason for this is a greater default minimum Trickle timer interval (t_{min}) in ContikiRPL than in TinyRPL and, consequently, a lower pace of DODAG information dissemination via DIO messages. Similarly to TinyRPL, the DODAG built by ContikiRPL remained stable, as shown in Fig. 3a: After the initial construction period, all nodes had a valid path to the root throughout the rest of the experiment.

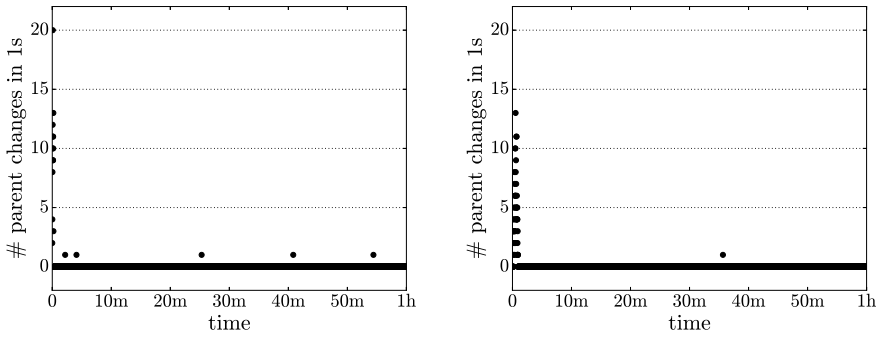
What is more, from Fig. 4, which plots the average DODAG rank of a node, it can be deduced that both implementations were able to construct optimal DODAGs. To explain, with the smallest allowed increase in rank per hop (*MinHopRankIncrease*) equal to 1 and the unit-disk links, a node's rank in an optimal DODAG would be equal to the node's Manhattan distance to the root plus 1. Consequently, the average rank of a node in an optimal DODAG would be equal to 11. It can be observed in the figure that the average node rank in the constructed DODAGs was indeed 11 (or at least close to 11 because of the limited resolution of the figure).

The metrics presented in the next three figures are used to analyze the stability of the DODAG after the initial construction period and the overhead of the control traffic in the period in which the DODAG should be stable.



(a) TinyRPL **(b)** ContikiRPL

Fig. 4 Average rank (unit disk, $r = 1$; MRHOF)

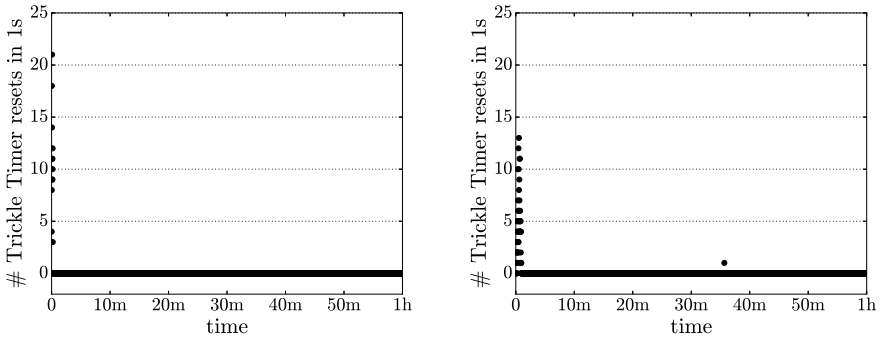


(a) TinyRPL **(b)** ContikiRPL

Fig. 5 Parent changes (unit disk, $r = 1$; duration = 1 h; MRHOF)

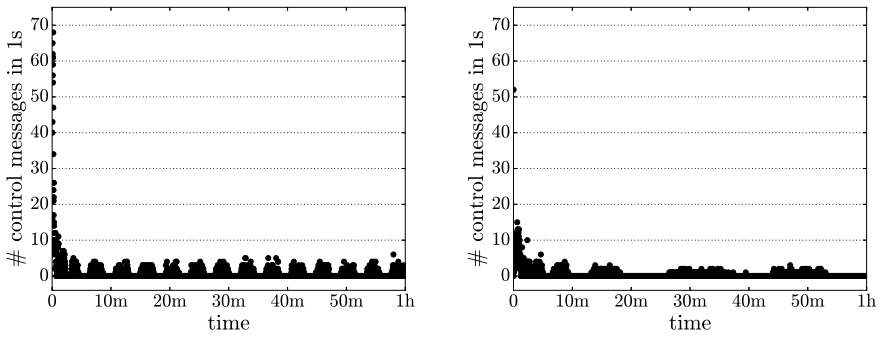
To start with, the plots in Fig. 5 present the number of preferred parent changes in an hour-long experimental run in the unit-disk topology with radius 1. Figure 6, in turn, shows the number of node Trickle timer resets in the same experiment. Recall that a node resets its Trickle timer whenever it observes an inconsistency or a vital change in the network. A large number of Trickle timer resets is thus an indicator of a DODAG's instability.

As can be observed in Fig. 5b, a DODAG constructed by ContikiRPL in the initial period stabilized after 1–2 min and changed only once throughout the whole experiment. This matches well Fig. 6b, which shows that all but one resets of node Trickle timers happened in the construction period. In contrast, TinyRPL did not build a final version of the DODAG in the initial period of the experiment. For this reason, a few preferred parent changes during the course of the experiment can be observed in Fig. 5a. Nevertheless, these few changes were not vital enough for the nodes to reset their Trickle timers. Consequently, there were no Trickle timer resets corresponding to these parent changes, as shown in Fig. 6a.



(a) TinyRPL **(b)** ContikiRPL

Fig. 6 Trickle timer resets (unit disk, $r = 1$; duration = 1 h; MRHOF)

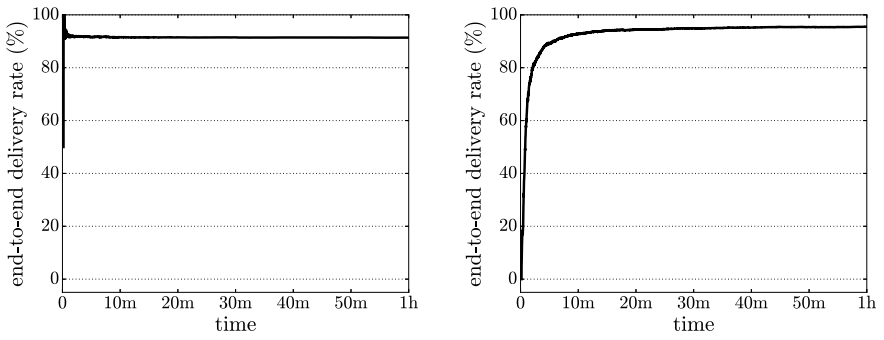


(a) TinyRPL **(b)** ContikiRPL

Fig. 7 Control messages (unit disk, $r = 1$; duration = 1 h; MRHOF)

Figure 7 illustrates the control traffic in the analyzed experiments. Control messages tracked in the plots are mainly DIO messages, scheduled by the nodes' Trickle timers. As a result, at the beginning of the experiment, when the Trickle periods were short, we can observe a large number of transmitted control messages. However, as soon as the DODAG stabilized and the periods of the node Trickle timers reached their highest values, the control traffic decreased and stayed low as long as the DODAG was stable.

As can be observed in both plots in Fig. 7, the control traffic was not uniform. In other words, periods of control message transmissions were interleaved with periods when there was no control traffic. There are two reasons for this phenomenon. First, since all nodes were booted roughly at the same time and almost immediately constructed the DODAG, their Trickle period starts were close in time. Second, the nodes scheduled their DIO messages in the second half of the Trickle period, thus leaving the aforementioned gaps in the control traffic. Moreover, the longest Trickle period for each implementation can be easily computed from the figures by summing



(a) TinyRPL **(b)** ContikiRPL

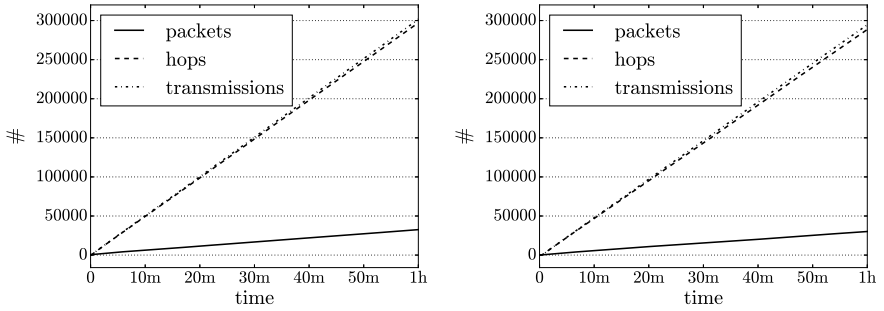
Fig. 8 End-to-end delivery rate (unit disk, $r = 1$; duration = 1 h; MRHOF)

the lengths of the longest gap and the following busy period. Accordingly, the longest Trickle period is about 4 min for TinyRPL and about 16 min for ContikiRPL, which matches their configurations. Consequently, when the DODAG is stable, out-of-the-box ContikiRPL generates less control traffic than out-of-the-box TinyRPL but this configuration can be changed.

The subsequent charts present results for metrics that were used to evaluate the reliability and resource consumption of the two implementations of RPL. The reliability was assessed by tracking *end-to-end delivery rate*. It was defined as the percent of all data packets generated by the nodes from the beginning of the experiment that were successfully delivered to the root. Resource consumption was in turn assessed by measuring *network traffic*, that is, the accumulated number of control and data packets generated by the nodes, the number of hops taken by those packets, as well as the number of physical radio transmissions of the packets. Since radio communication is typically the most resource-consuming activity of low-power wireless nodes, network traffic serves well as a proxy metric for resource consumption.

Figure 8 presents the end-to-end delivery rate for two-hour-long experiments evaluating both implementations. As can be observed in Fig. 8a, although the network did not suffer from any failures throughout the experiment, the end-to-end delivery rate for TinyRPL did not reach 100% but remained at the level of 90%. In other words, on average 1 data packet out of 10 was lost on its route toward the root and, consequently, did not reach the root. The reason for this was packet collisions caused by the small radio range of the nodes in the utilized topology and the high packet generation frequency ($T = 10$ s). On the other hand, due to the quick DODAG construction in TinyRPL, even packets generated by the nodes at the beginning of the experiment were successfully delivered to the root, which is visible in Fig. 8a as the high end-to-end delivery rate in the first seconds of the experiment.

In Fig. 8b, in turn, we can see that the final end-to-end delivery rate for ContikiRPL after one hour was slightly higher than that for TinyRPL. However, due to the longer DODAG construction period in ContikiRPL, the end-to-end delivery rate was low



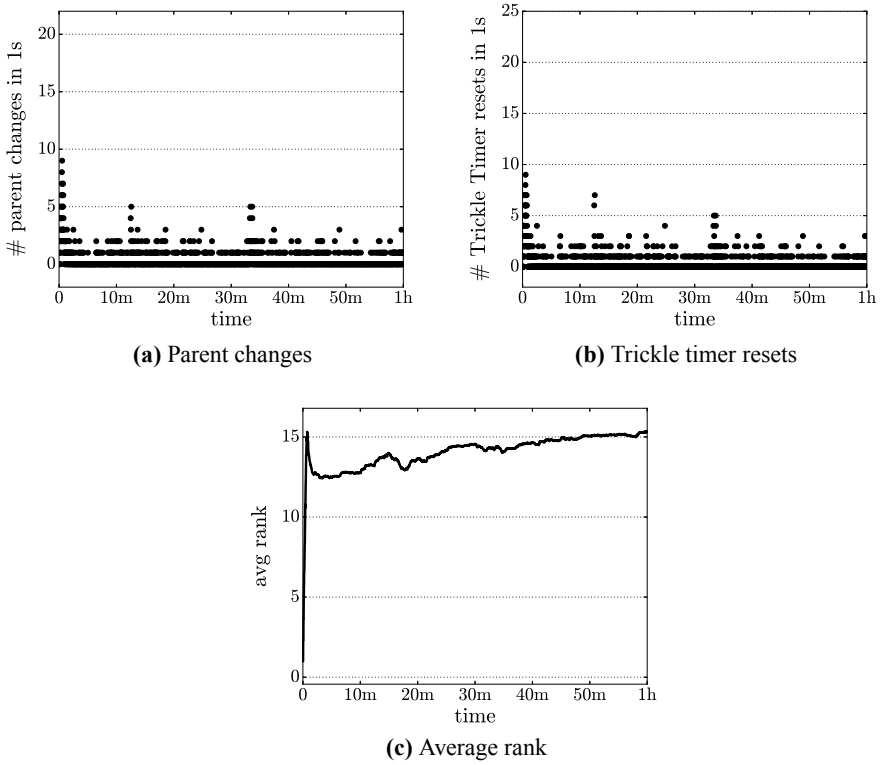
(a) TinyRPL **(b)** ContikiRPL

Fig. 9 Network traffic (unit disk, $r = 1$; duration = 1 h; MRHOF)

in the first seconds of the experiments and reached its highest value only after some period.

Figure 9, in turn, shows the network traffic in the two experiments. Packets, represented in the figure by a solid line, include both data packets generated every 10–20 seconds by each node and control packets carrying mainly DIO messages scheduled by the nodes' Trickle timers. Nevertheless, the number of control packets did not exceed 5% of the total number of packets. This also influenced the aggregate number of hops taken by the packets because the number of hops for a packet depends on the packet's type. In the analyzed configuration, it was always 1 for a control packet and on average 10 for a data packet. The transmissions in the figure, in turn, correspond to physical radio transmissions. Since control packets are broadcast and not acknowledged, the nodes never retransmit them. As a result, there was always 1 transmission for a single control packet. In contrast, due to transmission collisions, data packets were sometimes unacknowledged and required retransmitting. The maximum number of transmissions per data packet was limited by the value of the aforementioned configuration parameter, which was equal to 5. In practice, however, as can be observed in Fig. 9, retransmissions were occasional in both implementations and did not incur much overhead on resource consumption.

As mentioned previously, the analysis hitherto has not concerned the latest development version of ContikiRPL available from the repository at the time of this writing (i.e., the one from January 5, 2017). This is because that version of ContikiRPL does not construct a stable DODAG. More specifically, as can be observed in Fig. 10a, which plots the same experiment but with the latest version of ContikiRPL, a few nodes changed their preferred parent almost every second of the experiment. In contrast, as we have already shown in Fig. 5b, only one node changed its preferred parent after the construction phase in the experiment with the earlier version, selected for our analyses. The increased number of parent changes also resulted in an increased number of observed Trickle timer resets, as can be verified in Fig. 10b, and highly unstable ranks, visible in Fig. 10c. This, in turn, increased the accumulated network



(a) Parent changes

(b) Trickle timer resets

(c) Average rank

Fig. 10 ContikiRPL (latest version) (unit disk, $r = 1$; duration = 1 h; MRHOF)

traffic (not plotted), and hence, resource consumption. For these reasons, we settled on the earlier, yet more predictable and stable version of ContikiRPL.

5.1 Summary

To sum up, the experiments in the failure-free environment have shown that both analyzed implementations of RPL quickly construct stable DODAGs. Moreover, when used for routing, the constructed DODAGs deliver the majority of packets without wasting network resources. It can thus be concluded that the implementations, at least some of their versions, behave as expected in the failure-free environment. However, real-world networks are rarely free of failures. Therefore, in the next sections we proceed to failure-oriented experimental scenarios.

6 Experiments with Link Failures

We start our failure-oriented experiments with scenarios involving link failures. We consider a link as failed if every transmission over this link is lost, that is, it is not received by the target node. We evaluate how quickly the implementations react to link failures, depending on the chosen objective function and other configuration parameters. We also examine the consequences of link failures on the stability of the DODAG and reliability of packet forwarding.

Not all links in the network are equal. Depending on the network topology and the shape of the DODAG, some links are "important" in that packets from a significant fraction of nodes are forwarded through them to the DODAG root, whereas others are "unimportant" in that they are utilized by single nodes or not utilized at all. Moreover, the combinations of links that may fail are numerous. Since we are unable to test all link failure scenarios in this section, we analyze two selected ones: a failure of a single "important" link and a failure of a single "unimportant" link.

The experiments presented in this section lasted for two simulated hours; link failures occurred after the first hour. Such a timespan was chosen because it can be concluded from the previous section that an hour is enough for both implementations to construct DODAGs and stabilize.

6.1 Important Link Failure

In order to simulate a failure of an "important" link, we removed the link between nodes with identifiers 0 and 1 in the unit-disk topology with radius 1, as visualized in Fig. 11. The removed link was thus one of two links leading directly to the DODAG root and, as such, forwarded roughly half of data packets generated by all nodes.

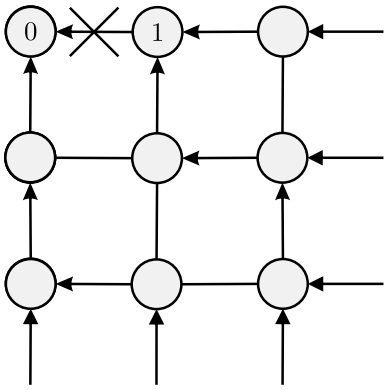
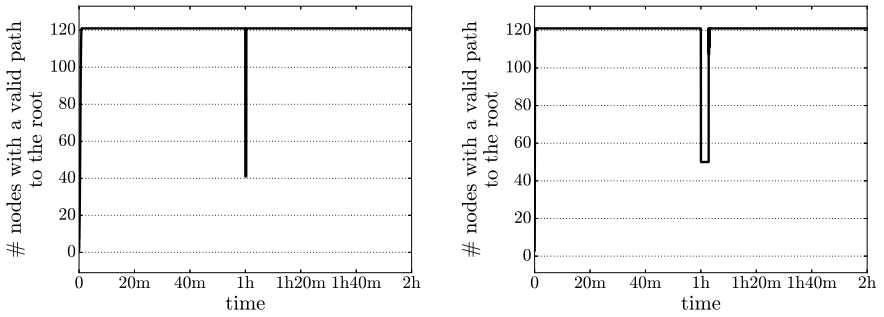


Fig. 11 Failure of an important link



(a) ContikiRPL **(b)** TinyRPL

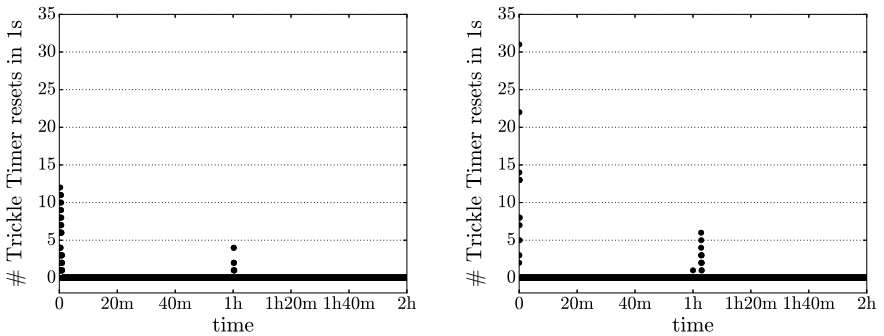
Fig. 12 Nodes with a valid path to the root (unit disk, $r = 1$; MRHOF)

6.1.1 MRHOF

Figures 12, 13 and 14 compare the behavior of the two implementations of RPL with MRHOF as the objective function in response to the analyzed link's failure.

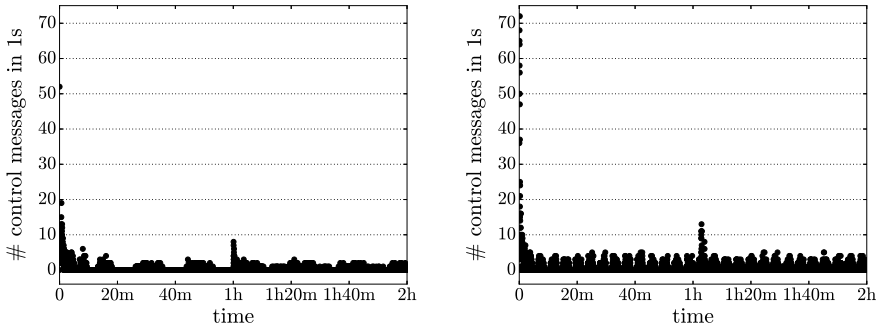
As can be observed in Fig. 12a, ContikiRPL reacted to the failure quickly: within seconds. In contrast, as visible in Fig. 12b, it took TinyRPL about 3 min to rebuild the DODAG, so that each node would again have a valid path to the root. This difference can be attributed mostly to the slightly different policies for resetting the Trickle timer in the two implementations.

More specifically, Fig. 13 presents the occurrences of Trickle timer resets caused by the failure. It can be observed in Fig. 13b that in TinyRPL the first node reset its Trickle timer immediately after the failure. However, the DODAG was not fully rebuilt until other nodes reset their timers 3 min later. In ContikiRPL, in turn, all additional resets of the node Trickle timers occurred immediately after the failure, as visible in Fig. 13a, and, therefore, the failure was handled more quickly.



(a) ContikiRPL **(b)** TinyRPL

Fig. 13 Trickle timer resets (unit disk, $r = 1$; MRHOF)



(a) ContikiRPL **(b)** TinyRPL

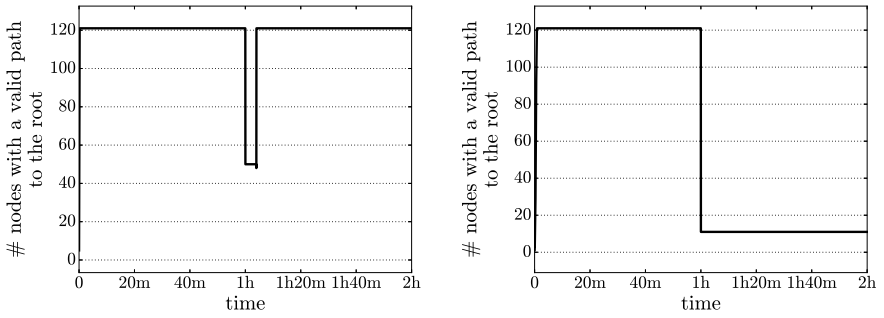
Fig. 14 Control messages (unit disk, $r = 1$; MRHOF)

Figure 14 depicts an increase in control traffic following the Trickle timer resets after the failure. In both implementations, an increase in the control traffic can indeed be observed. Nevertheless, it is not significant compared to the stable period in the experimental scenarios without failures, as presented in Fig. 7.

6.1.2 OF0

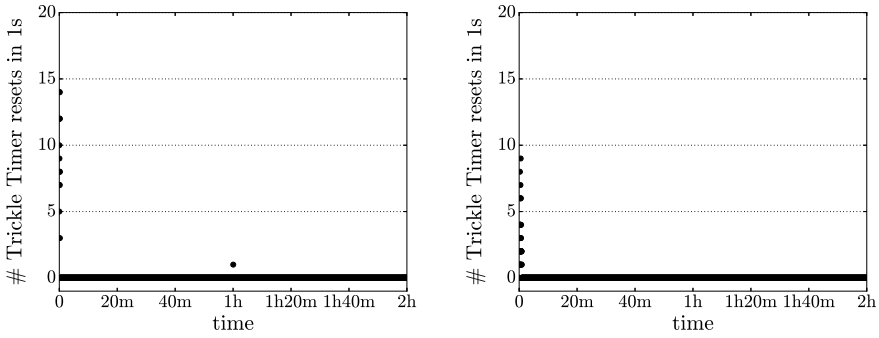
Although the reaction to the failure was not immediate in TinyRPL, both implementations managed to rebuild the DODAG when MRHOF was the objective function. Figures 15, 16 and 17 present the results of the same experiments but conducted with OF0 as the objective function.

It can be observed in Fig. 15a that with OF0 rebuilding all nodes' paths to the root took TinyRPL about 6 min, twice as long as with MRHOF. In contrast, it can be concluded from Fig. 15b that ContikiRPL with OF0 as the objective function did not



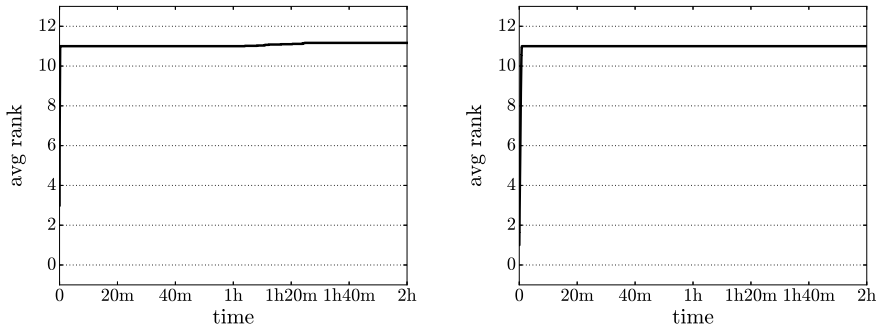
(a) TinyRPL **(b)** ContikiRPL

Fig. 15 Nodes with a valid path to the root (unit disk, $r = 1$; OF0)



(a) TinyRPL **(b)** ContikiRPL

Fig. 16 Trickle timer resets (unit disk, $r = 1$; OF0)

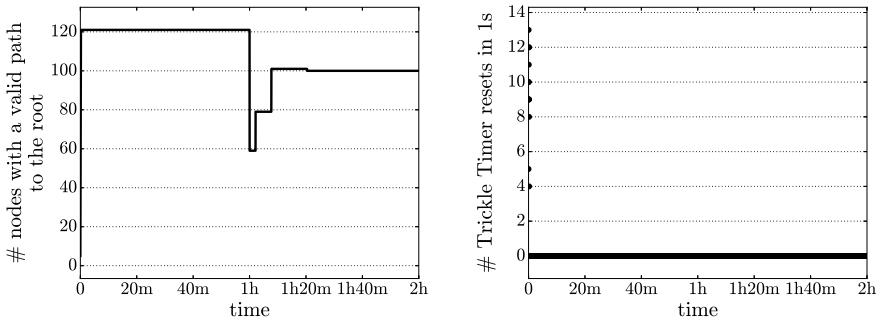


(a) TinyRPL **(b)** ContikiRPL

Fig. 17 Average rank (unit disk, $r = 1$; OF0)

manage to rebuild the DODAG at all. As a result, over 100 nodes, the nodes whose paths to the root contained the broken link, were not able to forward their packets to the destination throughout the rest of the experiment.

While the link failure in TinyRPL was followed by a Trickle timer reset, as can be observed in Fig. 16a, it turns out that ContikiRPL with OF0 as the objective function did not even react to the failure, not to mention recovering from it. In other words, as can be seen in Fig. 16b, no node in the network reset its Trickle timer in response to the failure. Moreover, it can be derived from Fig. 17b that since an average rank of a node in the network did not change as a result of the failure, then also the rank of the node with id 1 did not change. This, in turn, means that the node with id 1 did not notice for an hour that its link to the root was not working. The reason for this is that ContikiRPL does not estimate link metrics nor does it employ any neighbor unreachability detection mechanism when configured with OF0 as the objective function.



(a) Nodes with a valid path to the root **(b)** Trickle timer resets

Fig. 18 TinyRPL (unit disk, $r = 1$; OF0; max. retransmissions = 3)

Since ContikiRPL with OF0 as the objective function was not able to detect and react to the smallest possible failure, the failure of a single link, it would not be able to handle more complex failures either. As a result, we do not consider ContikiRPL in the configuration with OF0 in the remainder of the evaluation.

Although the analyzed experiments showed that, apart from ContikiRPL with OF0, all implementation-objective-function configurations were able to handle the failure and recover from it, this is not true for all possible values of configuration parameters. More specifically, decreasing the maximum number of retransmissions for a packet in TinyRPL made TinyRPL unable to rebuild the DODAG, irrespective of which objective function was used.

The results of an experiment with such a configuration are presented in Fig. 18. Similarly to ContikiRPL with OF0, not only did TinyRPL fail to rebuild the broken paths, which is visible in Fig. 18a, but also no node reacted to the failure by resetting its Trickle timer, as can be observed in Fig. 18b. We tracked this phenomenon to an emergent behavior of TinyRPL's code for neighbor unreachability detection but the explanation is out of the scope of this chapter.

6.2 Unimportant Link Failure

In order to simulate a failure of a link that is not responsible for forwarding a large part of all packets, the link between nodes with ids 0 and 1 in the unit-disk topology with radius 4 was removed, as depicted in Fig. 19.

Figure 20 presents the results of this experiment for both implementations with MRHOF as the objective function. As can be observed in Fig. 20, ContikiRPL reacted to the failure more quickly than TinyRPL, which matches the results of the experiments analyzed in the previous section. Nevertheless, the reaction times for both implementations were slightly higher than when the same link was subject to the failure in the previous experiment. This is because in this experiment the link was

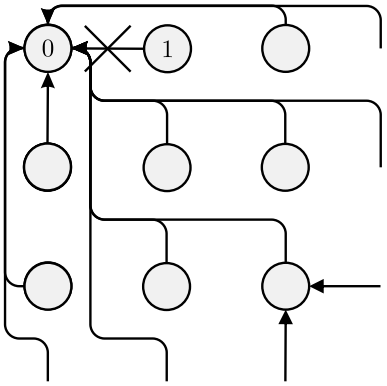
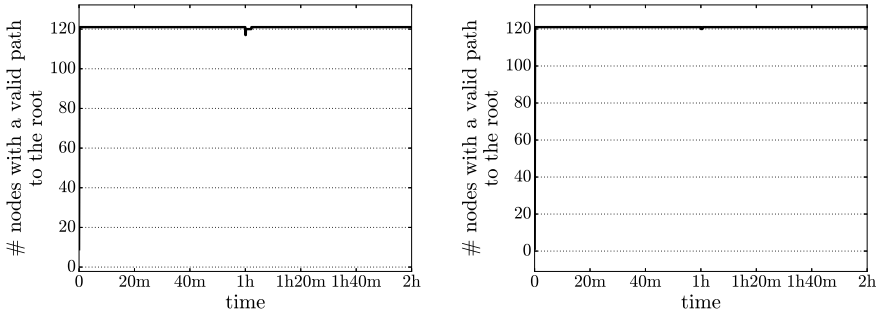


Fig. 19 An unimportant link failure



(a) TinyRPL **(b)** ContikiRPL

Fig. 20 Nodes with a valid path to the root (unit disk, $r = 4$; MRHOF)

more loaded—at the network level—than in the previous experiment: Since fewer packets failed to be forwarded over this link in a given time period, RPL was not able to detect the failure as fast as in the previous experiment.

The results for TinyRPL with OF0 did not differ significantly from the presented results. For this reason, their analysis is omitted here.

6.3 Summary

To sum up, the experiments with the simplest type of failure, affecting a single link, show that while the default configurations of the two implementations are by and large able to handle the failure, there exist out-of-the-box configurations for which this does not hold. What is more, in principle, it is not obvious why such configurations fail. This suggests that the two popular implementations of RPL may not yet be off-

the-shelf solutions; on the contrary, their adoption in real-world embedded systems may require expertise.

7 Experiments with Node Failures

As the next step, we analyze the behavior of the implementations under different types of node failures. Since any node failure can be simulated by correlated link failures, we consider only those configurations from the previous section in which an implementation was able to properly handle a link failure. Therefore, we do not examine ContikiRPL with OF0 in this section.

Except for one 3-h-long experiment, the experiments analyzed in this section lasted for 2 simulated hours. Similarly to the link failures in the experiments from the previous section, all node failures in the experiments from this section occurred after the first simulated hour.

7.1 Important Node Failure

Like links, nodes may be more or less "important," depending on the network topology and the shape of a DODAG. Drawing from the previous section, we focus only on the failures of "important" nodes, that is, ones that forward packets from a large fraction of other nodes. In order to test the implementations under the failure of such a node, we turned off one of the two root's neighbors, the node with id 1, in the unit-disk topology with radius 1, as shown in Fig. 21.

The results turned out to be very similar to the results of the experiments with failures of important links, analyzed in the previous section. To illustrate, Fig. 22

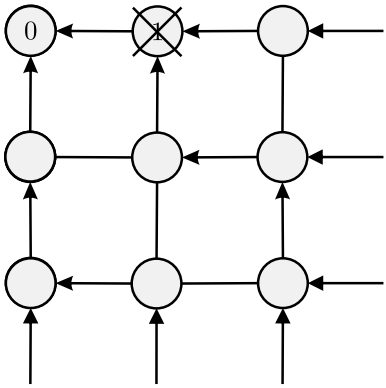
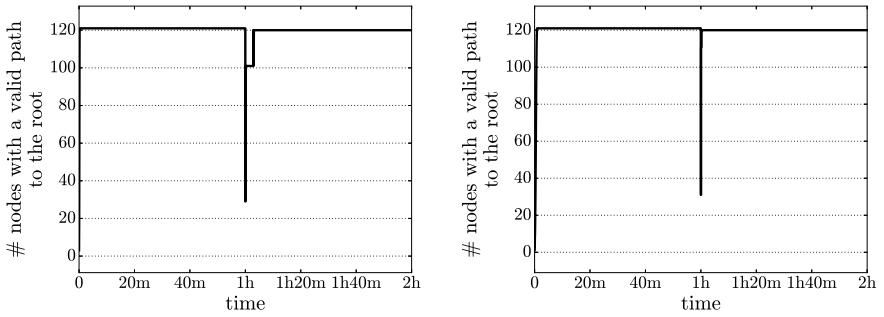


Fig. 21 Failure of an important node



(a) TinyRPL **(b)** ContikiRPL

Fig. 22 Nodes with a valid path to the root (unit disk, $r = 1$; MRHOF)

presents the number of nodes with a valid path to the DODAG root for TinyRPL and ContikiRPL with MRHOF as the objective function.

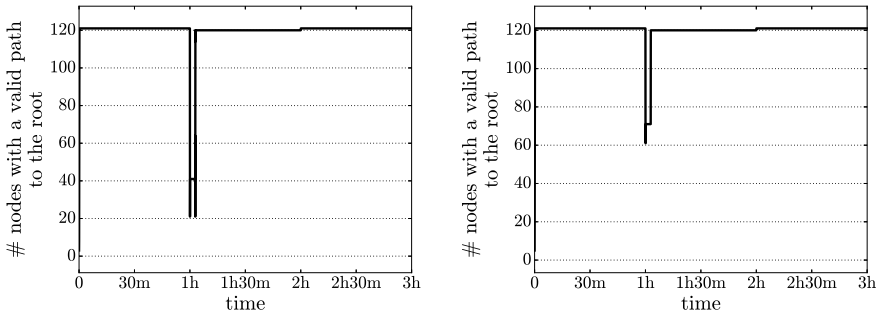
As can be observed in Fig. 22a, it took TinyRPL about 3 min to rebuild paths from all working nodes to the DODAG root after the failure. ContikiRPL, in turn, more quickly reacted to the failure and reconstructed the DODAG, as visible in Fig. 22b. Note that in both cases, the number of nodes with a valid path to the root after the failure did not reach the level it had before the failure; it was lower exactly by 1. However, since we assume that non-working nodes do not have a valid path to the root, this is an expected result. All in all, these results match precisely those obtained for the failure of the link between the nodes with ids 0 and 1 (cf. Fig. 12).

7.2 Node Failure and Recovery

Since failed nodes (and links) may recover, our next scenario aimed to evaluate the implementations' behavior in response to such a recovery. To this end, a 3-h-long experiment was conducted. Similarly to the previously described scenario, the node with id 1 was turned off after the first hour of the experiment (cf. Fig. 21). However, after the second hour the node was turned back on to see the implementations' reactions to the recovery. From the results of the previous experiment, we concluded that an hour is enough for both implementations to handle the failure.

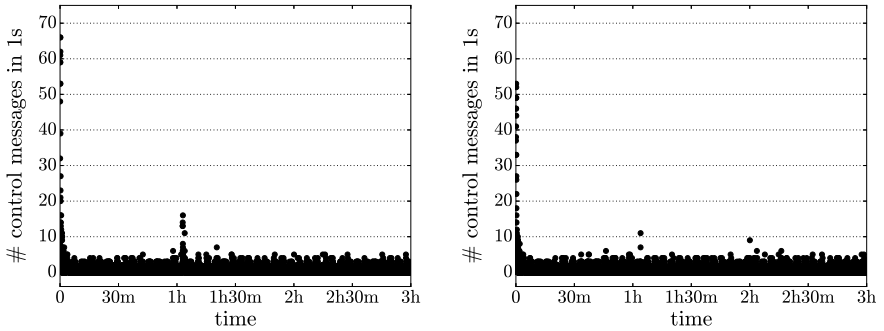
7.2.1 TinyRPL

First, we discuss the results for TinyRPL. Figure 23 presents the number of nodes with a valid path to the root throughout the experiment for MRHOF, Fig. 23a, and OF0, Fig. 23b. For both objective functions, a 3–5-min-long decrease in the analyzed metric occurred after the first hour of the experiment, immediately after the failure,



(a) MRHOF **(b)** OF0

Fig. 23 Nodes with a valid path to the root (TinyRPL; unit disk, $r = 1$)



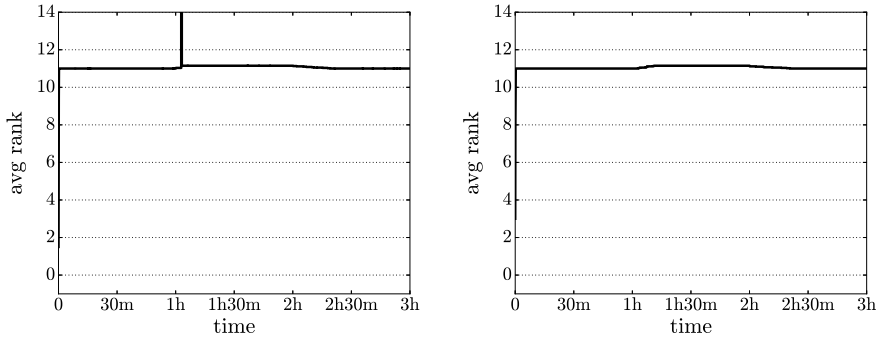
(a) MRHOF **(b)** OF0

Fig. 24 Control messages (TinyRPL; unit disk, $r = 1$)

which is in line with the previous results. After the second hour, in turn, a small increase could be noticed. The increase was caused by the node with id 1 reentering the network and brought the metric back to the value before the failure.

As can be observed in Fig. 24, neither the failure nor the recovery of the broken node resulted in a large increase in control traffic. Nevertheless, additional control messages transmitted as a result of the topology changes are visible in the plots for both objective functions.

An important result from this experiment is also presented in Fig. 25, showing the average rank of a node in the DODAG throughout the experiment. For both objective functions, an increase in the average rank can be observed after the first hour of the experiment, which can be attributed to detecting the failure. More specifically, nodes that selected node 1 as their preferred parent before the failure had to find an alternative preferred parent after the failure. Since the alternative parent had a higher rank than node 1, the nodes' ranks increased as a result of the change, and so did the average rank. In contrast, after the recovery of the failed node, TinyRPL brought the average rank back to the value before the failure, irrespective of which objective



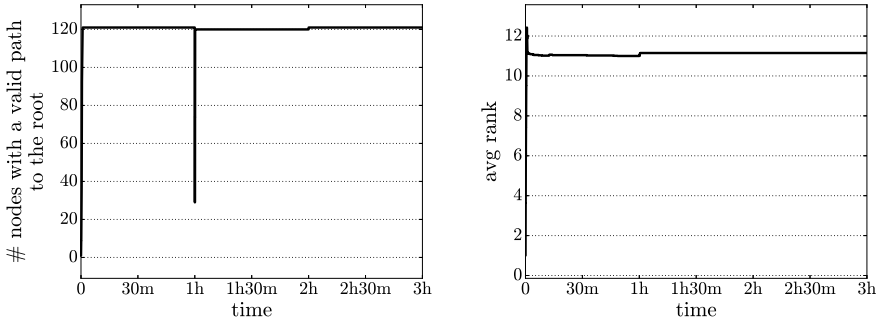
(a) MRHOF **(b)** OF0

Fig. 25 Average rank (TinyRPL; unit disk, $r = 1$)

function was used. This means that the DODAG restored by the implementation after the recovery was equally good as the one before the failure. Nevertheless, because of not generating much additional control traffic (cf. Fig. 24), this DODAG restoration process took some 20 min.

7.2.2 ContikiRPL

Figure 26 presents the results of the same experiment but for ContikiRPL with MRHOF. The plot of the number of nodes with a valid path to the DODAG root for ContikiRPL in Fig. 26a resembles the one for TinyRPL in Fig. 23a. However, a small difference can be observed in the plot for the average rank of a node in the DODAG for ContikiRPL in Fig. 26b and that for TinyRPL in Fig. 25a. Namely, in ContikiRPL, contrary to TinyRPL, the average rank did not change after node



(a) Nodes with a valid path to the root **(b)** Average rank

Fig. 26 ContikiRPL (unit disk, $r = 1$; MRHOF)

1 reentered the network. In particular, the routes built before the failure were not restored after the recovery of the failed node. As a result, some packets were routed through longer routes in the last hour of the experiment than in the first hour, thus generating slightly more network traffic than it was necessary. In other words, the DODAG restored by ContikiRPL was not as good as the one before the failure.

7.3 Correlated Node Failures

To complete the picture of node failures, we analyze a more complex scenario, that is, correlated failures of multiple nodes. Figure 27 presents the part of the unit-disk topology with radius 1 affected by the failure. The failure of three nodes, marked in the figure with crosses, occurred after the first hour of the experiment. Recall that the node with id 0 is the root of the DODAG. As a result, one of the broken nodes was the root's direct neighbor. This, together with the proximity of the broken nodes, makes the failure potentially difficult to handle.

Before we analyze the results of the experiment, let us focus on the node with identifier 2. Since we could observe in Sect. 5 that all analyzed implementations constructed an optimal DODAG, we can assume that the dashed links in Fig. 27 mark the path from the node with identifier 2 to the root before the failure. The bold solid links with arrows, in turn, determine the shortest possible path between these nodes after the failure. As a result of the failure, the length of the path from node 2 to the DODAG root increased so that the hop count became 6. Since the links in the analyzed topology are perfect and retransmissions rare, the ETX difference between the paths should not exceed 6 either. We thus consider the implementations with two configurations of *MaxRankIncrease*: one in which the nodes' rank growths caused by the failure should not be higher than *MaxRankIncrease* and another in which the growths would be higher for some nodes.

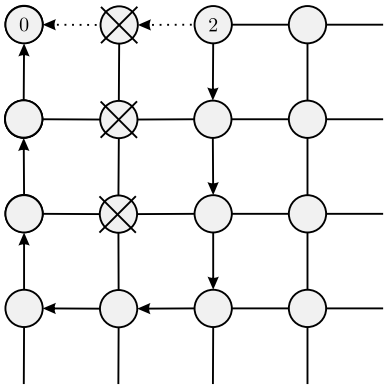


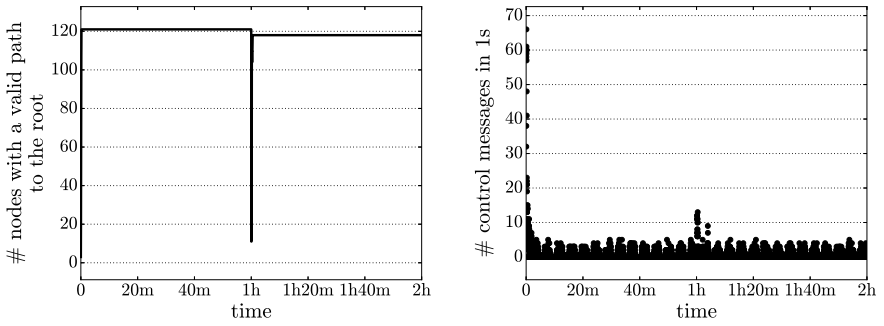
Fig. 27 Correlated node failures

7.3.1 New Rank Does Not Exceed *MaxRankIncrease*

With the default value of *MaxRankIncrease*, equal to 7, the rank growth caused by the failure for any node should not be higher. Consequently, for both OF0 and MRHOF, the implementations should reconstruct the DODAG after the failure.

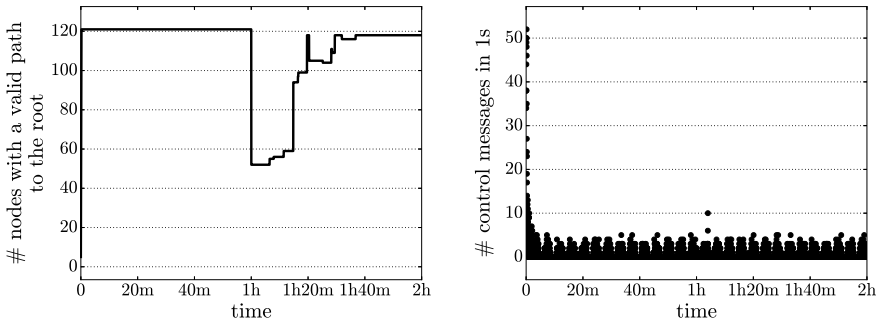
We start the analysis from TinyRPL. Figures 28 and 29 present TinyRPL's reaction to the failure when MRHOF and OF0, respectively, were used as the objective functions.

In the case of MRHOF, as can be observed in Fig. 28a, the number of nodes with a valid path to the root sharply decreased immediately after the failure, but then it quickly increased to the value of 118, which is the number of all nodes except for the three broken ones. In other words, TinyRPL with MRHOF detected the failure and reacted to it within seconds. The quick reaction, however, resulted in a significant increase in the control traffic, as visible in Fig. 28b and not observed in the previous experiments in this configuration.



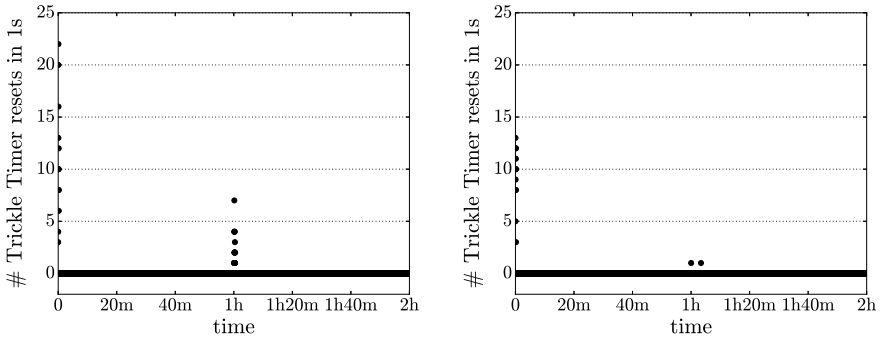
(a) Nodes with a valid path to the root **(b)** Control messages

Fig. 28 TinyRPL (unit disk, $r = 1$; MRHOF)



(a) Nodes with a valid path to the root **(b)** Control messages

Fig. 29 TinyRPL (unit disk, $r = 1$; OF0)



(a) MRHOF **(b)** OF0

Fig. 30 TinyRPL: Trickle timer resets (unit disk, $r = 1$)

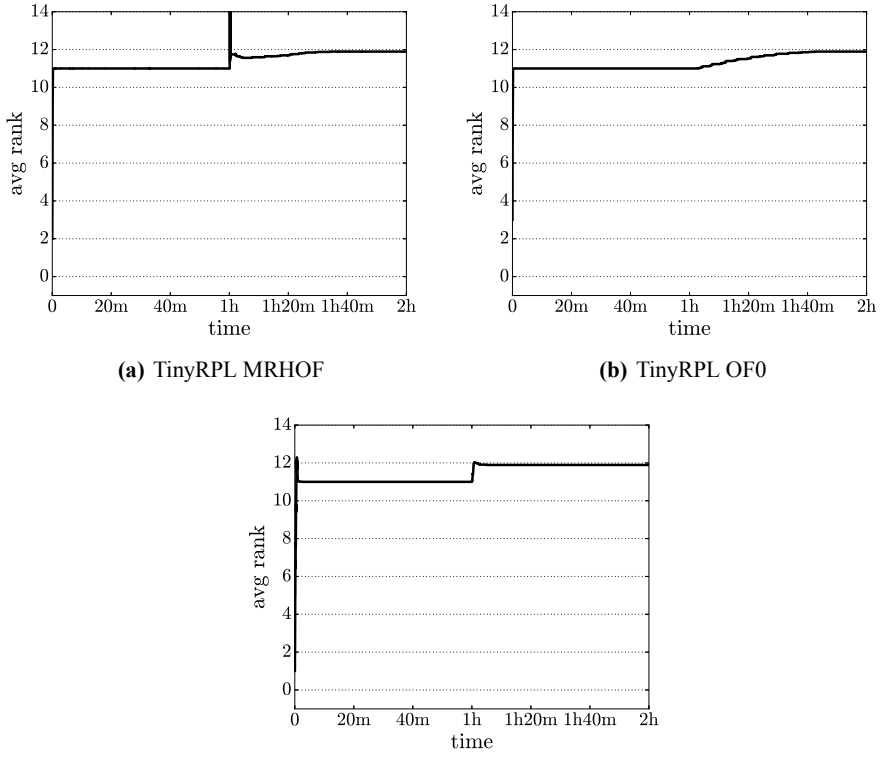
In contrast, as observed in Fig. 29a, the recovery process with OF0 as the objective function took TinyRPL over half an hour, far longer than with MRHOF. This resulted in a considerably lower total end-to-end packet delivery rate. On the other hand, the increase in control traffic presented in Fig. 29b is barely visible, and hence much lower when compared to that for MRHOF in Fig. 28b.

The reason for the observed differences in the two metrics in the experiments with MRHOF and OF0 lies in the number of times the nodes reset their Trickle timers in response to the failure. As can be observed in Fig. 30a, there were many Trickle timer resets when MRHOF was employed as the objective function. Consequently, the control messages traffic increased, the information on the failure was quickly disseminated in the network, and nodes could immediately handle the inconsistency. In contrast, when OF0 was used as the objective function, there were only two Trickle timer resets, as visible in Fig. 30b. They thus did not cause a significant increase in the control traffic. This, in turn, resulted in the slow propagation of DODAG information in the network and, consequently, long recovery time.

The results for ContikiRPL with MRHOF are similar to those for TinyRPL with MRHOF. Therefore, we do not present them here for brevity.

The experiments hitherto showed that the implementations were capable of rebuilding the DODAG, so that all working nodes could successfully forward packets to the root. However, let us examine how close the reconstructed DODAG was to the optimal one. To this end, Fig. 31 compares the average rank of a node in the DODAG after the failure.

It can be observed in Fig. 31a that although for TinyRPL with MRHOF there was a significant increase in the average rank immediately after the failure, the rank stabilized below 12 after several minutes. When OF0 was used, in turn, the average rank was slowly increasing for over half an hour after the failure, as visible in Fig. 31b. Nevertheless, as soon as the DODAG was fully reconstructed, the average rank also stabilized before reaching 12. In contrast, in Fig. 31c for ContikiRPL with MRHOF, a sharp increase in the average rank was followed by a smooth decrease. The average



(a) TinyRPL MRHOF

(b) TinyRPL OF0

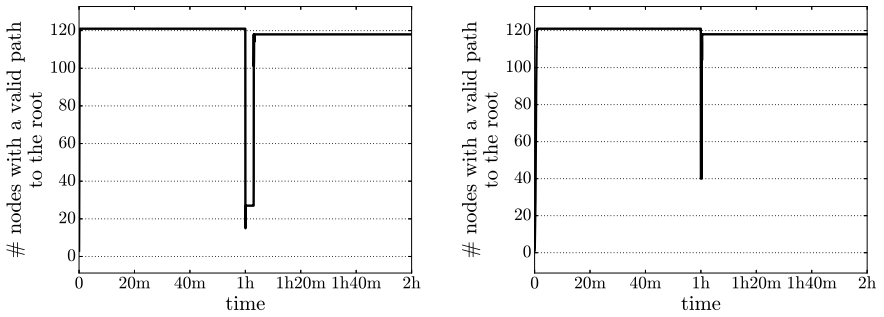
(c) ContikiRPL MRHOF

Fig. 31 Average rank (unit disk, $r = 1$)

rank, however, quickly dropped back below 12. It can thus be concluded that the DODAGs, once fully reconstructed, were equally close to optimal ones for both implementations.

7.3.2 New Rank Exceeds *MaxRankIncrease*

We reasoned at the beginning of this section that the increase in rank of the node with identifier 2 resulting from the failure is at least 6. Consequently, if we set *MaxRankIncrease* to 5 instead of 7, the node with identifier 2 should not be able to find a new valid path to the root after the failure because its rank would then exceed its minimum rank before the failure by more than *MaxRankIncrease*, which is forbidden according to RPL's specification. Figure 32 thus presents the number of nodes with a valid path to the root throughout the same experiment but with *MaxRankIncrease* set to 5.



(a) TinyRPL MRHOF **(b)** ContikiRPL MRHOF

Fig. 32 Nodes with a valid path to the root (unit disk, $r = 1$; *MaxRankIncrease* = 5)

As can be observed in the figure, in contrast to the specification, both TinyRPL and ContikiRPL managed to rebuild the paths for all 118 working nodes, including the node with identifier 2. This result is incorrect and is a consequence of the differences between RPL's specification and both analyzed implementations in tracking the smallest rank in the current DODAG version. In the next section, we give a more detailed explanation for these differences.

7.4 Summary

The experiments with node failures confirmed the conclusions gathered in the previous sections. While the two implementations of RPL correctly handle some simple and more complex scenarios, there exist situations where they do not behave as expected; on the contrary, their behavior is incorrect. Moreover, although the implementations seem to be able to recover from node failures, in some configurations the recovery takes a long time during which the network is not fully reliable.

8 Experiments with Network Partitions

As a final step, we evaluate the implementations in scenarios where some nodes get disconnected from the DODAG root as a result of failures. Recall that according to RPL's specification, a disconnected node should first detect that it does not have any path to the root. It should then discard its preferred parent, set its rank to infinity, and stop forwarding packets.

The experiments presented in this section lasted for 2 or 3 simulated hours, depending on a particular scenario; the failure occurred always after the first hour.

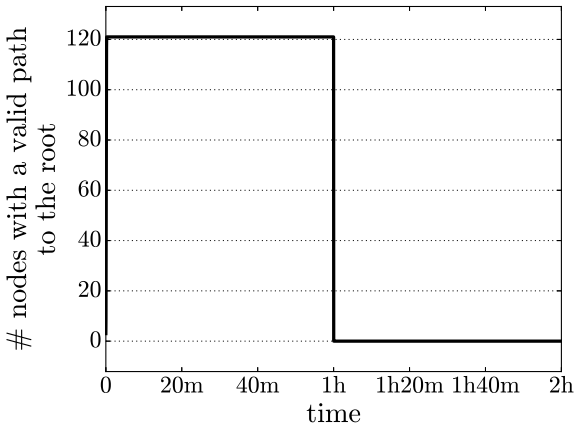


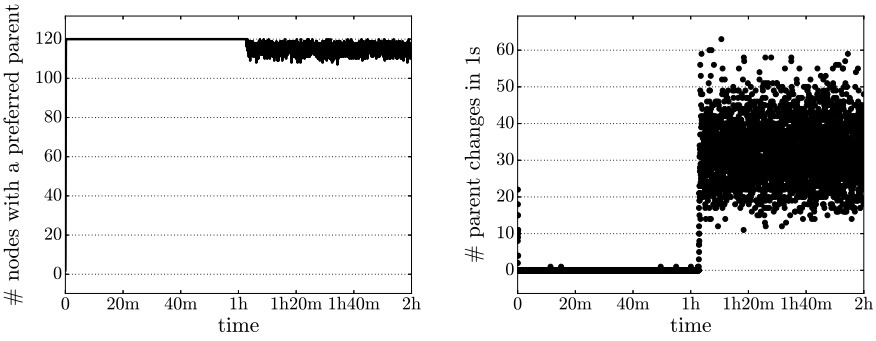
Fig. 33 Nodes with a valid path to the root under a failure of the root

8.1 Root Failure

One example of a failure resulting in a network partition is a failure of the DODAG root. As a result of such a failure, *all* nodes stop having a valid path to the root, which is visualized in Fig. 33. It is thus an extreme case of a network partition. We analyze it separately for each of the two implementations of RPL. For TinyRPL, we also consider the two objective functions, whereas for ContikiRPL—only MRHOF.

8.1.1 TinyRPL with MRHOF

Figures 34, 35 and 36 present the results of the experiment for TinyRPL with MRHOF as the objective function. It can be observed in Fig. 34a that, although no node had a



(a) Nodes with a preferred parent **(b)** Parent changes

Fig. 34 TinyRPL (unit disk, $r = 1$; MRHOF)

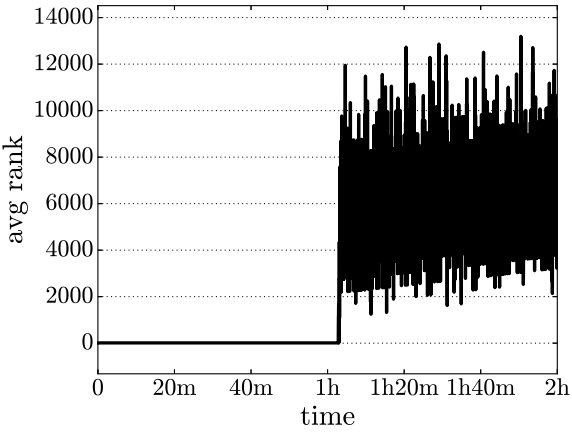
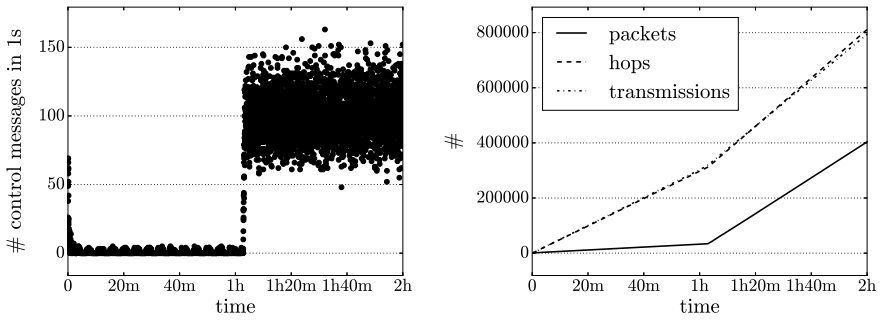


Fig. 35 TinyRPL: Average rank (unit disk, $r = 1$; MRHOF)

valid path to the root after the failure, the majority of nodes continuously had non-null preferred parents. Moreover, the nodes that discarded their preferred parents immediately selected new ones, thereby creating cycles in the DODAG. For this reason, a large number of parent changes can be observed in Fig. 34b after the failure. More specifically, the average number of parent changes in each second after the failure was five times the number of parent changes in the initial second during the DODAG construction. After the failure, the DODAG was thus highly unstable.

The average rank of the nodes in the DODAG is, in turn, presented in Fig. 35. After the failure, it fluctuated a lot. The sharp changes can be owed to the nodes changing their ranks to infinity, which is represented as value 65 535, and back again to finite values. Nevertheless, a constant growth in the average rank can be observed. In particular, the average rank growth exceeded the maximum allowed growth for a single node, as specified by *MaxRankIncrease*, without any visible reaction from the implementation.

The large number of parent changes after the failure was also accompanied by a huge increase in control traffic, which is observable in Fig. 36a. Namely, over 100 control messages per second were generated in the network after the failure, more than 10 times the number of data packets. This incurred a significant overhead on the accumulated traffic, and hence, global resource consumption. As shown in Fig. 36b, due to the increase in the control traffic, the total number of generated packets increased a few times. This, in turn, combined with forwarding data packets over cyclic routes, led to an increase of approximately 30% in the total number of transmissions. In conclusion, rather than reducing, TinyRPL actually increases the global network traffic and resource usage after a crash of the DODAG root.



(a) Control messages **(b)** Network traffic

Fig. 36 TinyRPL (unit disk, $r = 1$; MRHOF)

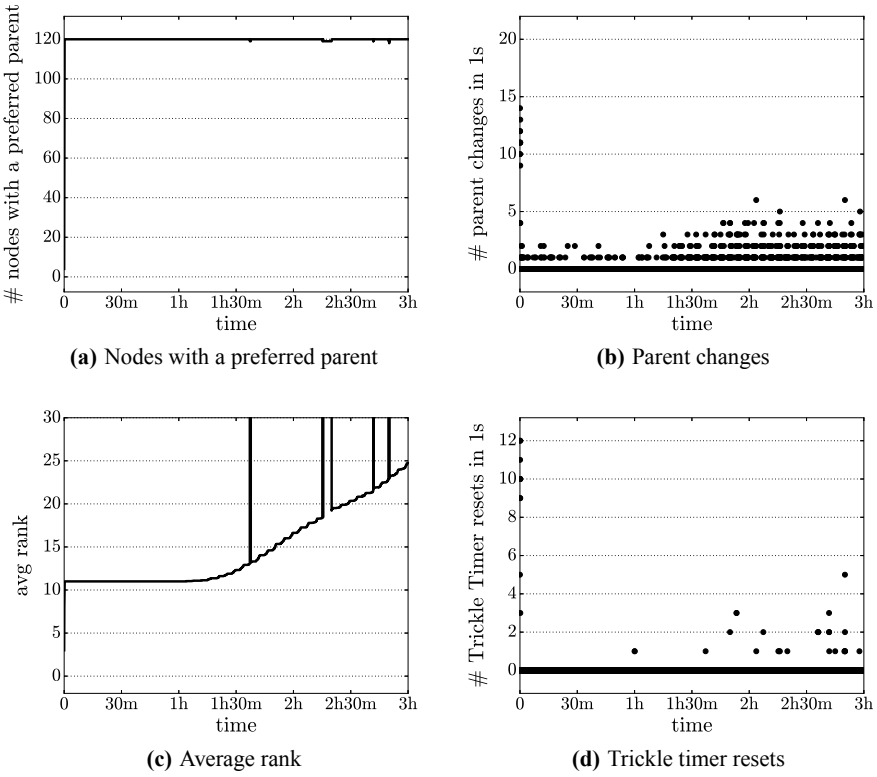
8.1.2 TinyRPL with OF0

Figures 37 and 38 present the results of a 3-h-long experiment for TinyRPL with OF0. As can be observed in Fig. 37b, the number of parent changes with OF0 was significantly lower than that with MRHOF, depicted in Fig. 34b. On the other hand, for the majority of time after the failure, all nodes had a non-null preferred parent, which is visible in Fig. 37a. In particular, no node discarded its preferred parent until almost an hour after the failure, and a node that finally did it, immediately selected a new preferred parent. In other words, the DODAG included cycles that were not broken for a long time. This is, however, the consequence of the fact that TinyRPL does not implement the mechanism for loop detection when OF0 is employed as the objective function. Moreover, it can be concluded that such mechanism is indeed necessary for RPL to behave efficiently in the presence of network changes.

Similarly to the results for MRHOF, a stable growth in the average rank, exceeding *MaxRankIncrease*, can be observed for OF0 in Fig. 37c. The growth rate is much lower in the case of OF0, though. This can be attributed to the slower reaction to network changes for TinyRPL with OF0 due to long Trickle timer intervals at the nodes. The intervals were long, in turn, as because of the lack of the loop detection mechanism, only a few nodes reset their Trickle timers in response to the failure, which can be verified in Fig. 37d.

Contrary to the experiments with MRHOF, an increase in the control traffic for OF0 is barely visible in Fig. 38a. This is because no loops were detected and therefore, the nodes did not reset their Trickle timers in response. Nevertheless, an about 25% growth in the number of hops and transmissions can be observed in Fig. 38b. The reason is that the nodes forwarded data packets through routes that contained the undetected cycles.

In conclusion, TinyRPL fails to correctly handle a crash of the DODAG root, irrespective of which objective function it is configured with. The main reason for this lies in TinyRPL's implementation of the enforcement mechanisms for the rank growth limit, described by *MaxRankIncrease*. As a reminder, according to RPL's



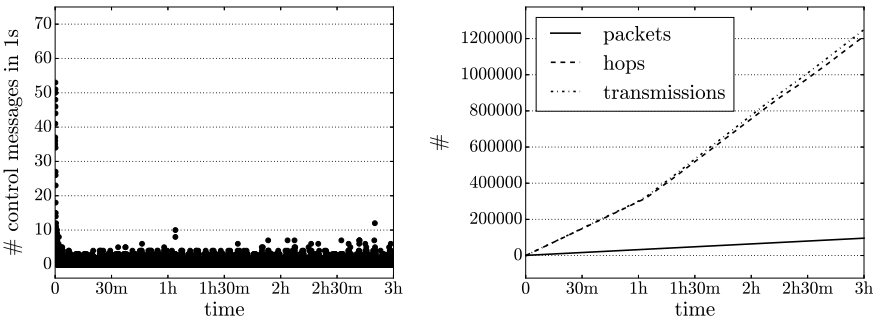
(a) Nodes with a preferred parent

(b) Parent changes

(c) Average rank

(d) Trickle timer resets

Fig. 37 TinyRPL (unit disk, $r = 1$; OF0)



(a) Control messages

(b) Network traffic

Fig. 38 TinyRPL (unit disk, $r = 1$; OF0)

specification, each node in the DODAG has to keep track of the smallest rank it has ever assigned in the current DODAG version. Whenever its rank starts to exceed the smallest rank by more than *MaxRankIncrease*, the node must conclude that the DODAG is broken, discard its preferred parent, and adopt an infinite rank. However, TinyRPL does not follow the specification in that the node keeps "forgetting" the smallest rank in the current DODAG version. In effect, the nodes' ranks may grow to infinity, as observed in our experiments.

8.1.3 ContikiRPL with MRHOF

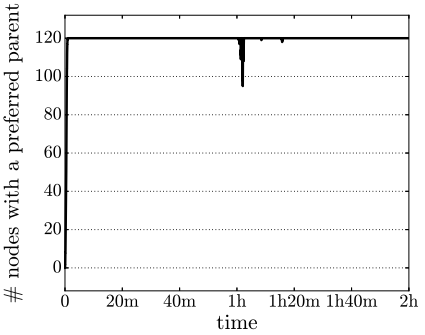
The results of the same experiment for ContikiRPL are presented in Figs. 39 and 40. As can be observed in Fig. 39a, the failure resulted in some nodes discarding their preferred parents and selecting new ones. Consequently, a growth in the number of parent changes is visible in Fig. 39b and in rank in Fig. 39c. Nevertheless, after a short period of changes, the DODAG stabilized again. More specifically, all nodes selected their preferred parents, the number of parent changes in one second dropped back to the level from before the failure and the average rank growth ceased. However, as visible in Fig. 39c, the average rank grew by more than 7 compared to the one before the failure, and hence, *MaxRankIncrease* was exceeded.

In Fig. 40a, in turn, it can be observed that although the failure resulted in an increase in control traffic, the increase did not last for more than a few minutes and hardly affected the accumulated network traffic, as can be verified in Fig. 40b. Consequently, contrary to TinyRPL, handling the crash of the DODAG root by ContikiRPL did not cause a significant increase in global resource usage.

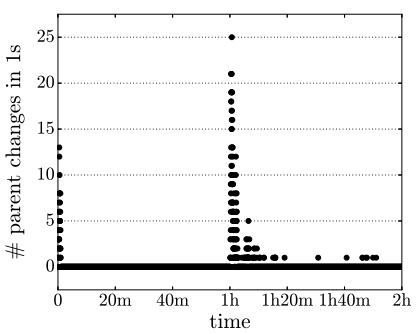
Nevertheless, while ContikiRPL's behavior is better than TinyRPL's, it is not fully correct either. In particular, despite the root being down, the nodes were still using their resources to forward generated data packets over the entire network to the root's neighbors, which, in turn, could not forward them further and were forced to drop them. Consequently, although the nodes did not generate extra traffic, the traffic they did generate was still far from optimal. Optimally, after the nodes detected the failure, they should have stopped forwarding any data packets as there was no route through which those packets could have been delivered to the root.

The reason for the observed behavior in ContikiRPL is the same as in the case of TinyRPL: improper management of the smallest rank assigned to a node in a given DODAG version. In the case of ContikiRPL, however, the incorrect changes to the value happen not virtually always but only occasionally. Nevertheless, in a network of a few tens of nodes, they are still a problem.

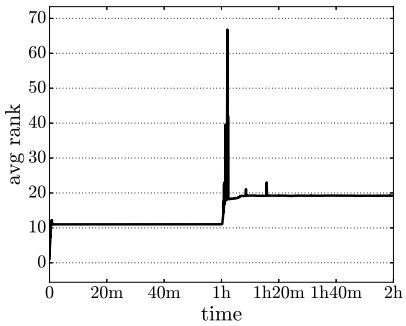
When it comes to the latest version of ContikiRPL, as of January 5, 2017, it behaves correctly in this failure scenario. More specifically, in response to the failure, all working nodes discarded their preferred parents, as visible in Fig. 41a, set their ranks to high values, as plotted in Fig. 41b, and stopped forwarding data packets, as can be observed in Fig. 41c. Consequently, there were only a few transmissions in reaction to the failure, which can be observed in Fig. 41c, and they were all transmissions of control messages. Note that the number of hops after the failure



(a) Nodes with a preferred parent

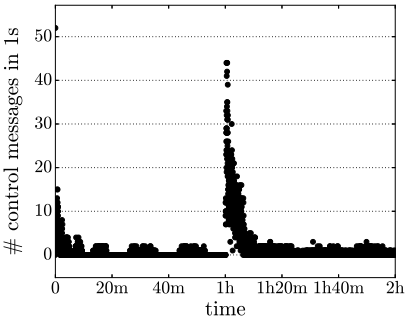


(b) Parent changes

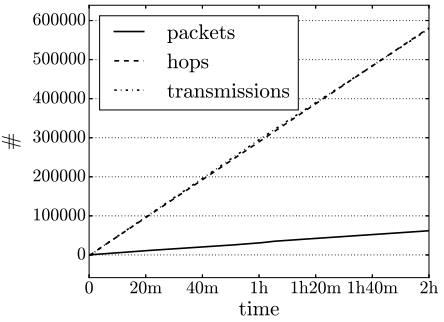


(c) Average rank

Fig. 39 ContikiRPL (unit disk, $r = 1$; MRHOF)

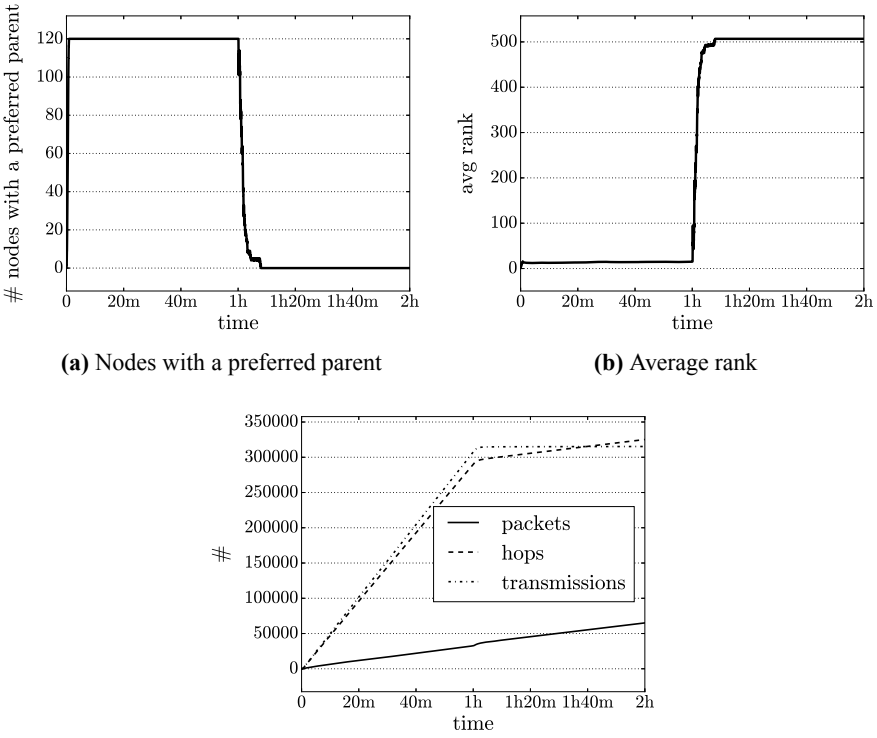


(a) Control messages



(b) Network traffic

Fig. 40 ContikiRPL (unit disk, $r = 1$; MRHOF)



(a) Nodes with a preferred parent

(b) Average rank

(c) Network traffic

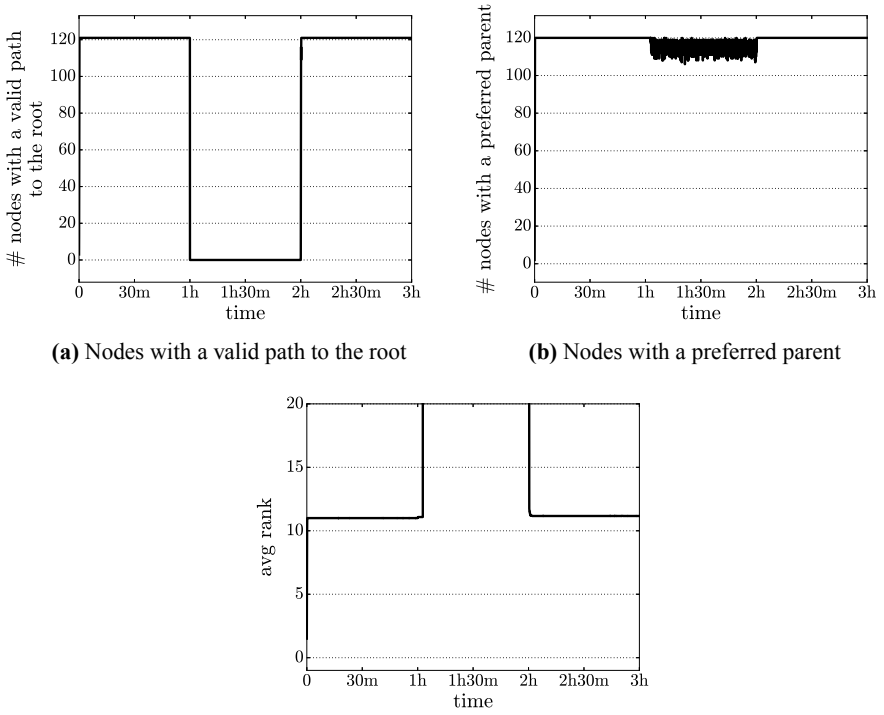
Fig. 41 ContikiRPL (latest version) (unit disk, $r = 1$; MRHOF)

increased faster than the number of transmissions. This was because a request from the test application to send a data packet counted as one hop even if the packet was not transmitted over the network because of, for example, a lack of the default route in the node's IPv6 routing table. Consequently, since the test application continued to generate data packets after the failure and RPL ceased to transmit them at some point, the growth rate of the number of hops exceeded that of the number of transmissions.

Nevertheless, recall that the DODAG constructed by the latest version of ContikiRPL is not stable. What is more, in contrast to the analyzed version, the new version fails in the next scenario, which we believe is even more important from a practical perspective.

8.2 Root Failure and Recovery

Hitherto, we have shown that neither TinyRPL nor ContikiRPL correctly handles a crash of the DODAG root. Our subsequent experiments aim to examine whether



(a) Nodes with a valid path to the root

(b) Nodes with a preferred parent

(c) Average rank

Fig. 42 TinyRPL (unit disk, $r = 1$; MRHOF)

the implementations manage to rebuild a DODAG and return to a stable state when the root recovers from its failure. The experiments lasted for 3 h. The root failure occurred after the first hour of each experiment, whereas the recovery after the second hour.

8.2.1 TinyRPL

In the case of TinyRPL with MRHOF, as can be observed in Fig. 42a, b, the DODAG was reconstructed immediately after the root had recovered. Moreover, since the average rank quickly returned to the level from before the failure, as shown in Fig. 42c, it can be concluded that the reconstructed DODAG was as good as the initial one.

Figure 43 presents the network traffic throughout the experiment. Although an increase in the number of generated packets and performed transmissions can be observed between the failure and the recovery, the metrics returned to the values from before the failure as soon as the root recovered.

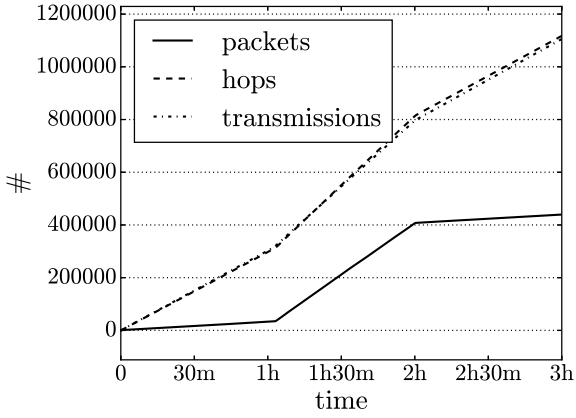


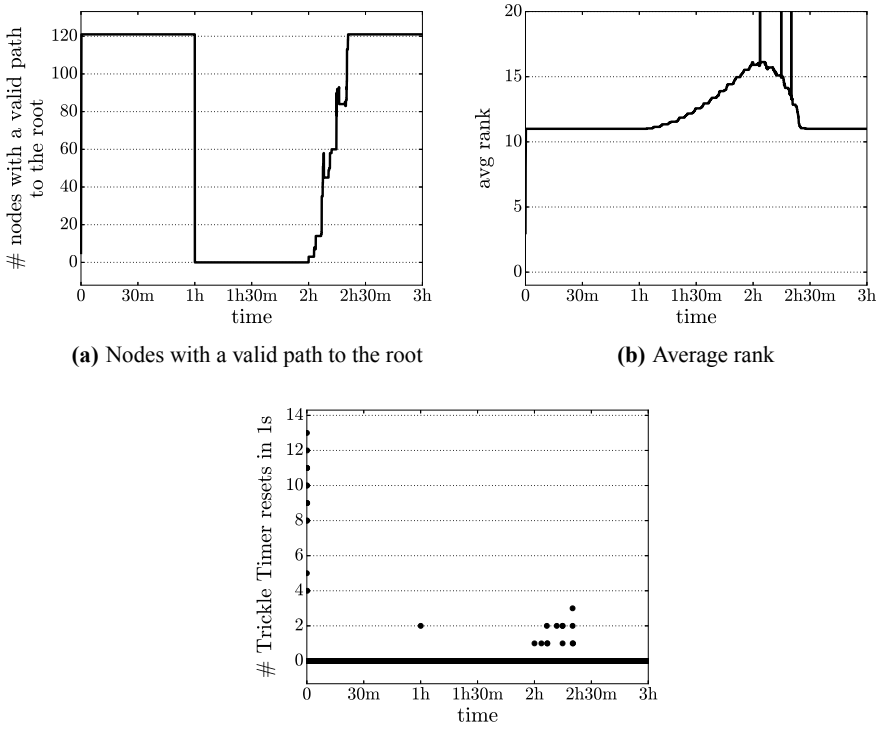
Fig. 43 TinyRPL: Network traffic (unit disk, r=1; MRHOF)

Figure 44 presents the results of the same experiment, but with OF0 as the objective function. Although TinyRPL with OF0 managed to rebuild the DODAG and bring the average rank of a node back to the value from before the failure, the reconstruction process took the implementation significantly longer than when MRHOF was used, about 20 min instead of a few seconds. Again, this can be attributed to the low number of nodes resetting their Trickle timers in response to network changes when OF0 is employed as the objective function, which can be verified in Fig. 44c.

8.2.2 ContikiRPL

In ContikiRPL, in turn, all nodes rebuilt their paths to the root within seconds after the recovery, as shown in Fig. 45a. It can be observed in Fig. 45b, however, that it took the implementation more than half an hour to bring the average node's rank back to the value before the failure. The reason for this is that TinyRPL and ContikiRPL manage their parent sets in a different way, the details of which we omit here for brevity.

Finally, recall that while the analyzed version of ContikiRPL does not react correctly to the root failure, its latest version does. Let us thus check whether the latest version also correctly handles the DODAG root's recovery. As can be observed in Fig. 46, the latest version of ContikiRPL did not reconstruct the DODAG after the root had recovered from the failure. More specifically, neither did any non-root node select a preferred parent, which can be verified in Fig. 46a, nor did it adopt a low rank, which is visible in Fig. 46b in the third hour of the experiment. It can be thus concluded that the implementation is not capable of recovering from a failure of the DODAG root. Again, the reason lies in yet another method of maintaining node parent sets in the new version of ContikiRPL compared to the previous versions.

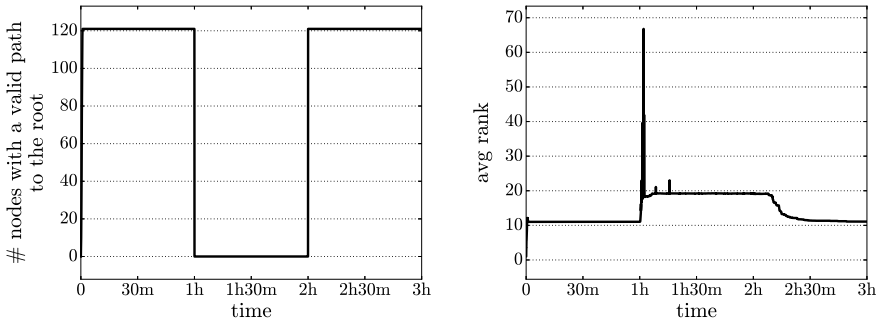


(a) Nodes with a valid path to the root

(b) Average rank

(c) Trickle timer resets

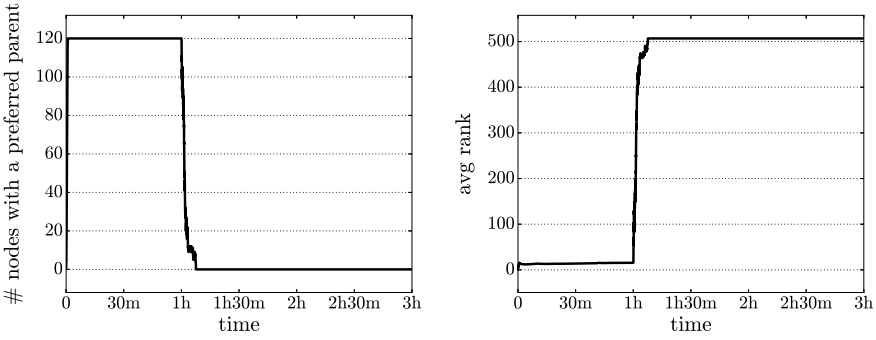
Fig. 44 TinyRPL (unit disk, $r = 1$; OF0)



(a) Nodes with a valid path to the root

(b) Average rank

Fig. 45 ContikiRPL (unit disk, $r = 1$; MRHOF)



(a) Nodes with a preferred parent **(b)** Average rank

Fig. 46 ContikiRPL (latest version) (unit disk, $r = 1$; MRHOF)

8.3 Summary

The DODAG root failure scenario analyzed in this section was the most extreme case of a network partition. The results for other scenarios with network partitions are analogous to the presented ones. For this reason, we omit them here.

All in all, the results for scenarios involving network partitions are unfavorable for both analyzed implementations of RPL. Neither TinyRPL nor ContikiRPL correctly reacts to such failures: nodes keep forwarding data packets to the root even though all their paths to the root are broken, thereby unnecessarily wasting resources. In some cases, the failure additionally results in a huge increase in network traffic and, consequently, resource consumption. This, in turn, may be a major obstacle to deploying RPL in real-world embedded systems. Moreover, it turns out that the failure-handling behavior of the implementations may vary drastically between versions, which complicates deployments even more.

9 Conclusions

To sum up, the following general conclusion can be drawn from our dependability-oriented experiments with the two popular implementations of RPL. When a network is not subject to any failures, both implementations behave as expected: Their control traffic volume gradually decreases to the minimal values and remains so, which corresponds to RPL's stable state. However, as soon as failures are introduced into the links and/or nodes, the implementations' behavior starts to diverge from the desirable one, sometimes with grave consequences for the network.

More specifically, it turns out that although the implementations are capable of handling simple link or node failures in the majority of the evaluated parameter settings, there exist configurations in which such failures are not even detected, not to

mention proper handling. The results for more complex failures, notably those leading to network partitions, are even more concerning. In both TinyRPL and ContikiRPL, nodes keep forwarding packets to the root node even if all their routing paths are broken. An effect of this inability to conclude that a major failure has taken place is that the nodes unnecessarily waste precious resources on transmitting packets that are never delivered to the root. In TinyRPL, the control traffic actually explodes after the failure, which could drain the energy of typical battery-powered nodes in a few days rather than months or years; in ContikiRPL, the increase is less pronounced. In any case, however, such futile transmissions combined with the lack of automatic nodes' reaction to the failure may give the network administrators an impression that everything functions properly, which may delay detecting the failure even by the human administrators. This, in turn, may be particularly problematic if the root node is an actuator that controls some important equipment. All in all, the two popular implementations of RPL are simply not robust against failures.

This is in stark contrast to the requirements of many LLN-based embedded systems. Because of the characteristics of LLNs, failures of both links and nodes are not uncommon. Consequently, it is crucial for RPL's implementations to handle such failures in a correct and efficient manner. The evaluated implementations thus have to be fixed before they can be utilized in real-world systems in which dependability is important. However, the fixing process may not be straightforward. The example of the different versions of ContikiRPL shows that addressing problems in one usage scenario may have unpredictable consequences in others. In other words, it may not be easy to determine whether a given change to the implementation is actually a "fix." We may thus need better methods of verifying the compliance of an implementation with RPL's specification. What is more, it is not clear whether fixing the implementations is possible without changing RPL's specification itself. It may well be that changes to the specification or novel algorithms [18] are necessary to improve failure handling. Importantly, we may also need formal methods for proving the correctness of the core protocol and such algorithms.

Acknowledgements This work was supported by the National Center for Research and Development (NCBR) in Poland under grant no. LIDER/434/L-6/14/NCBR/2015. K. Iwanicki was additionally supported by the Polish Ministry of Science and Higher Education with a scholarship for outstanding young scientists.

References

1. Boubekeur, F., Blin, L., Leone, R., Medagliani, P.: Bounding degrees on RPL. In: Q2SWinet '15: Proceedings of the 11th ACM Symposium on QoS and Security for Wireless and Mobile Networks, pp. 123–130. ACM (2015). https://doi.org/10.1145/2815317.2815339
2. Brachman, A.: RPL objective function impact on LLNs topology and performance. In: 13th International Conference on Internet of Things, Smart Spaces, and Next Generation Networking, NEW2AN 2013 and 6th Conference, ruSMART 2013, St. Petersburg, Russia, 28–30 August 2013, Proceedings, pp. 340–351. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40316-3_30

3. Clausen, T., Herberg, U., Philipp, M.: A critical evaluation of the IPv6 routing protocol for low power and lossy networks (RPL). In: 2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 365–372. IEEE (2011). https://doi.org/10.1109/WiMOB.2011.6085374
4. Conta, A., Gupta, M.: Internet control message protocol (ICMPv6) for the Internet protocol version 6 (IPv6) specification. RFC 4443 (2006). https://doi.org/10.17487/RFC4443
5. De Couto, D.S.J., Aguayo, D., Bicket, J., Morris, R.: A high-throughput path metric for multi-hop wireless routing. In: MobiCom '03: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, pp. 134–146. ACM (2003). https://doi.org/10.1145/938985.939000
6. Dunkels, A., Gronvall, B., Voigt, T.: Contiki—a lightweight and flexible operating system for tiny networked sensors. In: 29th Annual IEEE International Conference on Local Computer Networks, pp. 455–462. IEEE (2004). https://doi.org/10.1109/LCN.2004.38
7. Duquennoy, S., Landsiedel, O., Voigt, T.: Let the tree bloom: scalable opportunistic routing with ORPL. In: SenSys '13: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, pp. 2:1–2:14. ACM (2013). https://doi.org/10.1145/2517351.2517369
8. Fonseca, R., Ratnasamy, S., Zhao, J., Ee, C.T., Culler, D., Shenker, S., Stoica, I.: Beacon vector routing: scalable point-to-point routing in wireless sensornets. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, NSDI '05, pp. 329–342. USENIX Association (2005)
9. Frey, H., Pind, K.: Dynamic source routing versus greedy routing in a testbed sensor network deployment. In: Proceedings of the 6th European Conference on Wireless Sensor Networks, EWSN '09, pp. 86–101. Springer (2009). https://doi.org/10.1007/978-3-642-00224-3_6
10. Gaddour, O., Koubâa, A.: RPL in a nutshell: a survey. Comput. Netw. **56**(14), 3163–3178 (2012). https://doi.org/10.1016/j.comnet.2012.06.016
11. Gaddour, O., Koubâa, A., Chaudhry, S., Tezeghdanti, M., Chaari, R., Abid, M.: Simulation and performance evaluation of DAG construction with RPL. In: Third International Conference on Communications and Networking, pp. 1–8. IEEE (2012). https://doi.org/10.1109/ComNet.2012.6217747
12. Gnawali, O., Levis, P.: The minimum rank with hysteresis objective function. RFC 6719 (2012). https://doi.org/10.17487/RFC6719
13. Han, D., Gnawali, O.: Performance of RPL under wireless interference. IEEE Commun. Mag. **51**(12), 137–143 (2013). https://doi.org/10.1109/MCOM.2013.6685769
14. Heurtefeux, K., Menouar, H., AbuAli, N.: Experimental evaluation of a routing protocol for WSNs: RPL robustness under study. In: 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 491–498 (2013). https://doi.org/10.1109/WiMOB.2013.6673404
15. Hui, J., Vasseur, J.P.: The routing protocol for low-power and lossy networks (RPL) option for carrying RPL information in data-plane datagrams. RFC 6553 (2012). https://doi.org/10.17487/RFC6553
16. Iova, O., Theoleyre, F., Noel, T.: Stability and efficiency of RPL under realistic conditions in wireless sensor networks. In: 2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), pp. 2098–2102. IEEE (2013). https://doi.org/10.1109/PIMRC.2013.6666490
17. Istomin, T., Kiraly, C., Picco, G.P.: Is RPL ready for actuation? A comparative evaluation in a smart city scenario. In: 12th European Conference on Wireless Sensor Networks, EWSN 2015, Porto, Portugal, 9–11 February 2015, Proceedings, pp. 291–299. Springer International Publishing (2015). https://doi.org/10.1007/978-3-319-15582-1_22
18. Iwanicki, K.: RNFD: routing-layer detection of DODAG (root) node failures in low-power wireless networks. In: 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), pp. 13:1–13:12. IEEE (2016). https://doi.org/10.1109/IPSN.2016.7460720
19. Iwanicki, K., van Steen, M.: On hierarchical routing in wireless sensor networks. In: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks, IPSN '09, pp. 133–144. IEEE Computer Society (2009)

20. Iwanicki, K., van Steen, M.: A case for hierarchical routing in low-power wireless embedded networks. ACM Trans. Sens. Netw. **8**(3):25:1–25:34 (2012). https://doi.org/10.1145/2240092.2240099
21. Khelifi, N., Kammoun, W., Youssef, H.: Efficiency of the RPL repair mechanisms for low power and lossy networks. In: 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 98–103. IEEE (2014). https://doi.org/10.1109/IWCMC.2014.6906339
22. Kim, Y.J., Govindan, R., Karp, B., Shenker, S.: Geographic routing made practical. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, NSDI '05, pp. 217–230. USENIX Association (2005)
23. Ko, J., Eriksson, J., Tsiftes, N., Dawson-Haggerty, S., Terzis, A., Dunkels, A., Culler, D.: ContikiRPL and TinyRPL: happy together. In: Proceedings of the Workshop on Extending the Internet to Low power and Lossy Networks (IP+SN 2011) (2011)
24. Korte, K.D., Sehgal, A., Schönwälder, J.: A study of the RPL repair process using ContikiRPL. In: Dependable Networks and Services: 6th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, AIMS 2012, Luxembourg, Luxembourg, 4–8 June 2012, Proceedings, pp. 50–61. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30633-4_8
25. Levis, P., Lee, N., Welsh, M., Culler, D.: TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In: SenSys '03: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, pp. 126–137. ACM (2003). https://doi.org/10.1145/958491.958506
26. Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., Culler, D.: TinyOS: An Operating System for Sensor Networks, pp. 115–148. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/3-540-27139-2_7
27. Levis, P., Clausen, T., Hui, J., Gnawali, O., Jo, K.: The Trickle algorithm. RFC 6206 (2011). https://doi.org/10.17487/RFC6206
28. Mao, Y., Wang, F., Qiu, L., Lam, S.S., Smith, J.M.: S4: small state and small stretch routing protocol for large wireless sensor networks. In: Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation, NSDI '07, pp. 8–8. USENIX Association, Berkeley, CA, USA (2007)
29. Mohammad, M., Guo, X., Chan, M.C.: Oppcast: exploiting spatial and channel diversity for robust data collection in urban environments. In: 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), pp. 19:1–19:12. IEEE (2016). https://doi.org/10.1109/IPSN.2016.7460681
30. Narten, T., Nordmark, E., Simpson, W., Soliman, H.: Neighbor discovery for IP version 6 (IPv6). RFC 4861 (2007). https://doi.org/10.17487/RFC4861
31. Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., Voigt, T.: Cross-level sensor network simulation with COOJA. In: Proceedings of the 2006 31st IEEE Conference on Local Computer Networks, pp. 641–648. IEEE (2006). https://doi.org/10.1109/LCN.2006.322172
32. Radoi, I.E., Shenoy, A., Arvind, D.: Evaluation of routing protocols for Internet-enabled wireless sensor networks. In: ICWMC 2012: The Eighth International Conference on Wireless and Mobile Communications (2012)
33. Thubert, P.: Objective function zero for the routing protocol for low-power and lossy networks (RPL). RFC 6552 (2012). https://doi.org/10.17487/RFC6552
34. Tripathi, J., de Oliveira, J.C., Vasseur, J.P.: A performance evaluation study of RPL: routing protocol for low power and lossy networks. In: 2010 44th Annual Conference on Information Sciences and Systems (CISS), pp. 1–6. IEEE (2010). https://doi.org/10.1109/CISS.2010.5464820
35. Vasseur, J.P., Kim, M., Pister, K., Dejean, N., Barthel, D.: Routing metrics used for path calculation in low-power and lossy networks. RFC 6551 (2012). https://doi.org/10.17487/RFC6551
36. Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J.P., Alexander, R.: RPL: IPv6 routing protocol for low-power and lossy networks. RFC 6550 (2012). https://doi.org/10.17487/RFC6550

Part II
Deployment and Coverage

On the Optimization of WSN Deployment for Sensing Physical Phenomena: Applications to Urban Air Pollution Monitoring

Ahmed Boubrima, Walid Bechkit and Hervé Rivano

Abstract Wireless sensor networks (WSN) are widely used in environmental applications where the aim is to sense a physical phenomenon. Air pollution is one of the main physical phenomena that still need to be studied and characterized. Using WSN for air pollution monitoring usually targets two main applications: regular mapping and the detection of high pollution concentrations. Both of these applications need a careful deployment of sensors in order to get better knowledge of air pollution while ensuring a minimal deployment cost. In this chapter, we present three formulations of the deployment issue of sensor and sink nodes based on integer linear programming modeling (ILP) while tackling the two main applications of air pollution monitoring. In the first ILP model, we target the regular mapping of air pollution based on predicted pollution maps. In the second and third ILP model, we target the detection of pollution threshold crossings. The second model is based on predicted pollution maps as in the first one, whereas the third one is based on pollution emission inventory which describes pollution sources and their emission rates. In addition to the constraints of pollution coverage, we also ensure that the deployed networks are connected and their financial deployment cost is minimized. We perform extensive simulations in order to analyze the performance of the proposed models in terms of coverage and connectivity results.

Keywords Wireless sensor networks (WSN) · Deployment · Coverage Connectivity · Air pollution monitoring · Air pollution mapping · Air pollution detection · Air pollution prediction · Atmospheric dispersion modeling

1 Introduction

Wireless sensor networks (WSN) are widely used in environmental applications where the aim is to sense a physical phenomenon such as temperature, humidity, air pollution. In this context of application, the use of WSN allows to understand the

A. Boubrima · W. Bechkit (✉) · H. Rivano
Univ Lyon, Inria, INSA Lyon, CITI, 69621 Villeurbanne, France
e-mail: walid.bechkit@insa-lyon.fr

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_4

variations of the phenomenon over the monitoring region and therefore be able to take adequate decisions regarding the impact of the phenomenon. Air pollution is one of the main physical phenomena that still need to be studied and characterized because it highly depends on other phenomena such as temperature and wind variations. Moreover, air pollution affects human health dramatically according to the World Health Organization (WHO), which states that exposure to air pollution is accountable to seven million casualties in 2012 [1].

Most of current air pollution monitoring is performed using measuring stations which are accurate but massive, inflexible, and expensive [2]. An alternative—or complementary—solution would be to use electrochemical sensor networks, which are smaller and cheaper and have a reasonable measurement quality [3, 4]. The main advantages of a WSN infrastructure for air pollution monitoring are to obtain a finer spatiotemporal granularity of measurements, thanks to the resulting lighter installation and operational costs [5].

Using WSN for air pollution monitoring usually targets two main applications: (i) the regular mapping where the objective is to construct pollution maps based on the data gathered by sensors; and (ii) the detection of high pollution concentrations where the objective is to ensure that data gathered by sensors allow to detect the threshold crossings of air pollution [6]. Both of these applications need a careful deployment of sensors in order to get better knowledge of air pollution while ensuring a minimal deployment cost.

In this chapter, we present three formulations of the deployment issue of sensor and sink nodes based on integer linear programming modeling (ILP) while tackling the two main applications of air pollution monitoring. In the first ILP model, we target the regular mapping of air pollution while constraining the positions of sensors by the quality of pollution maps which will be constructed by data gathered by sensors once deployed. Since gathering data cannot be done before deployment, we base our first model on predicted pollution maps. In the second and third ILP models, we target the detection of pollution threshold crossings while ensuring that sensors will be deployed at positions where high concentrations may occur. The second model is based on predicted pollution maps as in the first one, whereas the third one is based on pollution emission inventory which describes pollution sources and their emission rates. In addition to the constraints of pollution coverage, we also ensure that the deployed networks are connected so that every sensor is capable of sending its data to a sink node directly or through relay nodes. In all the three models, the objective is to minimize the financial cost of WSN deployment which is usually equivalent to minimizing the number of nodes to deploy. For each ILP model, we show a proof of concept performed on a real-world data set. We also analyze the performance of the proposed models in terms of coverage and connectivity results.

The remainder of this chapter is organized as follows. We first review the related works on the WSN deployment issue in Sect. 2. Then, we present and analyze the literature methods used for air pollution prediction in Sect. 3. Next, we present the formulation and the simulation results of our three ILP models in Sects. 4, 5, and 6. Finally, we provide some future directions and conclude the chapter in Sects. 7 and 8.

2 WSN Deployment

The deployment issue of wireless sensor networks has been addressed extensively in the literature where several mathematical models, optimal algorithms, and near-optimal heuristics have been proposed [7]. The problem consists in determining the optimal positions of sensors and sinks so as to cover the environment and ensure the network connectivity while optimizing an objective function such as the deployment cost or the coverage quality [8]. The network is said connected if each sensor can communicate information to at least one sink. Coverage requires that each point of interest is monitored with one or more sensor nodes. In addition to coverage and connectivity, the main issues targeted in the literature are energy consumption, network lifetime, and the network deployment cost.

This section identifies what lacks in the literature and motivates the need of application-aware deployment models for air pollution monitoring where the aim is to place sensors in order to optimize their number and the quality of pollution coverage that results from the data gathered by sensors once deployed. We present the related works based on their coverage definition while identifying their formulation of connectivity and network lifetime.

Existing deployment approaches can be considered as either event-aware like the works presented in [9–16] or correlation-aware like the works presented in [17–23]. In the first case, a sensor is assumed to have a detection range, usually circular, within which the sensor is capable of detecting any event that may happen. The second class of deployment approaches is based on the correlation that sensor measurements may present in order to select the minimum number of sensor nodes.

2.1 Event-Aware Deployment Methods

Chakrabarty et al. [9] represent the deployment region as a grid of points and propose a nonlinear formulation for minimizing the deployment cost of sensors while ensuring complete coverage of the deployment region. Then, they apply some transformations to linearize the first model and obtain an ILP formulation. The authors formulate coverage based on the distance between the different points of the deployment field. Each sensor has a circular detection area, which defines the points that the sensor can cover. Unfortunately, this measure of coverage is inadequate to the air pollution monitoring using electrochemical sensors since a sensor positioned at a point A cannot cover a neighboring point B if there is a difference between pollution concentrations at the two points.

Altinel et al. [10] proposed another formulation based on the set cover problem, which is equivalent to the aforesaid model but less complex. They also extend their formulation to take into account the probabilistic sensing of sensor nodes while assuming that a node is able to cover a given point with a certain predefined probability. Despite that, this new formulation is still generic since the dependency between

the errors of the deployed sensors is not considered. However, this has to be taken into account when doing air pollution estimation.

Chang et al. [11] proposed to use data fusion in the definition of coverage in order to take into account the collaborative detection of targets. They based in their work on a probabilistic sensing model to define the probability of target detection and the false alarm rate. Then, they formulated a nonconvex optimization problem minimizing the number of nodes under coverage constraints. They presented resolution algorithms and showed that the obtained solutions are near-optimal and hence very close to the optimal ones. Still, this work considers the existence of a detection range.

In addition to coverage formulation, the authors of [12] formulate connectivity based on the flow problem while assuming that sensors generate flow units in the network and verify if sinks are able to recover them. Another connectivity formulation has been introduced in [13] where authors base on an assignment approach. They introduce in their ILP formulation new variables to define the communication paths between sensors and sinks. However, this model involves more variables than the one based on the flow problem and is therefore more complex.

In [14], authors study the trade-off between coverage, connectivity, and energy consumption. They formulate the problem as an ILP model and then propose a multi-objective approach to optimize coverage, the network lifetime, and the deployment cost while maintaining the network connectivity.

In some other works, authors suppose that a set of connected sensors that ensure coverage are already deployed and propose integer programming formulations to find optimal sinks locations and sensors-to-sinks routes. Authors in [15] evaluate firstly the shortest path cost between each sensor and potential sink location using the Dijkstra algorithm. Two main metrics were proposed to compute shortest paths: energy cost and financial cost. In the second case, the proposed ILP model aims to find the optimal sinks positions while minimizing the financial cost of the sinks deployment and the sensors-to-sinks routes. Two other formulations based on flows were proposed in [16] where authors present a single commodity flow and multi-commodity flow formulations. However, they show that the integer programming model presented in [15] is better. Moreover, they propose and test good heuristics for this latter.

2.2 Correlation-Aware Deployment Methods

In [17], Roy et al. tackled the problem of finding the most informative locations of sensors for monitoring environmental applications. They assume the existence of a set of data snapshots characterizing the phenomenon to monitor. Then, they formulate the problem to find the best locations of sensors in order to reconstruct the data of the whole phenomenon with a required precision. Two optimization models are proposed to handle both stationary and non-stationary fields. An iterative resolution algorithm is proposed to solve the two deployment problems. Unfortunately, this work is based on a strong assumption; that is, sensor measurements are perfect and do not present

any drift, which is not the case of pollution sensors where sensed values may be different than pollution ground truth values.

In [18], Krause et al. tackle the same problem based on the assumption that the variations of the phenomenon are Gaussian. They also assume a pre-deployment phase allowing to gather data that can be used to characterize the phenomenon. In order to select the best positions of sensors, they use the concept of mutual information in order to define the quality of a given topology. After the formulation of the problem, they use the sub-modularity of mutual information to define a polynomial algorithm. This work considers only coverage and is extended in [19] to take into account the cost of connectivity where the links qualities are assumed to be Gaussian. Since air pollution is not necessarily Gaussian, this work does not fit our application case.

In [20], authors focus on the specific application of soil moisture sensing while assuming the Gaussian distribution of the phenomenon. They first show that their application case presents some particular characteristics that can be used to design application-aware deployment schemes. Based on these characteristics, they propose a clustering approach that allows them to divide the deployment problem into a set of disjoint subproblems. They perform extensive simulations while considering different correlation-aware deployment algorithms in the resolution of the subproblems. They conclude based on the obtained results that their clustering approach allows them to get solutions that are as good as the global solution of the deployment problem.

In [21], authors consider a different context where some sensors are already deployed for the monitoring of the Columbia River. They perform field estimation based on sensor measurements in addition to a simulation model, which is commonly known as data assimilation. Their work is mainly designed for the determination of sensor nodes whose information is redundant and therefore can be turned off to optimize the network lifetime. Their approach can be also used to determine the optimal positions of sensors that should be added to the network in order to improve coverage quality. However, the proposed approach cannot be used to perform a first deployment of sensors without considering existing nodes or a trial deployment phase.

As in [21], authors of [22] consider an already deployed sensor network and propose an algorithm to define a sensing topology to select active sensors and turn off the others. They estimate the variations of the phenomenon in an online way to decide whether a sensor is to keep active or not. In contrary to this work, in our case, the sensing locations have to be chosen in an offline way since the selection of sensing points is performed before the network deployment.

The mathematical characteristics of the correlation-aware deployment problem have been studied by Ranieri et al. [23] while considering a generic form. A greedy heuristic is proposed to solve the problem. They perform extensive simulations to show that their algorithm is capable of solving the problem in a short time compared to the existing heuristics while providing a near-optimal solution.

2.3 Discussion

Even if the recent works take into account network constraints like connectivity and energy consumption, all coverage formulations either assume that sensors have a given detection range, which is the case of event-aware methods, or the assumption is instead made on the distribution of sensor measurements, which is the case of correlation-aware methods. Novel application-aware deployment methods have been recently proposed to consider the characteristics of the application case in the design of the deployment approach; examples include the work of [24] on wind monitoring and the work of [25] for water pollution monitoring. Following the same direction, we propose in the remainder of this chapter to consider the context of air pollution monitoring and present appropriate formulations of coverage in addition to network connectivity.

3 Air Pollution Prediction

The three realistic formulations presented in this chapter are based on the nature of the phenomenon. This is ensured by integrating in the ILP models the variations of pollutants over the deployment region. The variations can be estimated using the so-called air pollution prediction. In this section, we introduce the main methods used in this latter.

Air pollution prediction allows to estimate pollution concentrations at a given point in the environment based on some measurements performed in the neighborhood of the point in question, weather conditions, the urban characteristics of the environment, and the characteristics of the pollution sources located within the deployment region, also known as the emission inventory [26]. Three major methods are used in air pollution prediction: atmospheric dispersion, interpolation, and land-use regression [27].

3.1 Atmospheric Dispersion-Based Methods

Atmospheric dispersion models take as input locations of pollution sources, the pollutant emission rate of each pollution source, and meteorological data in order to measure the pollutant concentration at a given location [27]. The obtained concentrations can then be calibrated using the measurements of sensors. The theoretical study of pollution atmospheric dispersion is mainly based on fluid mechanics theory [28]. For the sake of clarity, we present in this section only steady state dispersion, in particular Gaussian dispersion. The basic Gaussian model estimates the concentrations of a pollutant gas released by a pointwise pollution source in a free space environment [29]. The estimated value C (g/m^3) at a measurement location (x, y, z) is given by Formula (1). Table 1 details the parameters of the model.

Table 1 Parameters of the Gaussian dispersion model

Measurement location	
x	Downwind distance from the pollution source (m)
y	Crosswind distance from the pollution source (m)
z	Hight (m)
Pollution emission parameters	
h_s	Pollutant source height (m)
Q	Mass flow rate at the emission point (g/s)
V	Volumetric flow rate at the emission point (m^3/s)
T_s	Pollutant temperature at the emission point (K)
Weather	
T	Ambient air temperature (K)
V_w	Wind velocity (m/s)
D_w	Wind direction (degree)
Constants	
a_y, b_y	Horizontal dispersion coefficients
a_z, b_z	Vertical dispersion coefficients
g	Gravity constant (9.8 m/s^2)

$$C(x, y, z) = \frac{Q}{2\pi V_w \sigma_y \sigma_z} e^{-\frac{y^2}{2\sigma_y}} (e^{-\frac{(z-H)^2}{2\sigma_y}} + e^{-\frac{(z+H)^2}{2\sigma_y}}) \tag{1}$$

$$\Delta h = \frac{1,6 \cdot F^{1/3} \cdot x^{2/3}}{V_w} \tag{2}$$

The pollution source is located at the point $(0, 0, h_s)$, and the measurement point location is given according to a 3D coordinate system where the x-axis is oriented in the wind direction D_w. Parameters σ_y and σ_z describe the stability of the atmosphere and can be approximated using Briggs formulas: $\sigma_y = a_y \cdot |x|^{b_y}$ and $\sigma_z = a_z \cdot |x|^{b_z}$. The parameter H, which represents the pollutant effective release height, is equal to the sum of the pollutant source height h_s and the plume rise Δh. The pollution plume is located above the pollution source, and Δh is the vertical distance between the source and the center of the pollution plume. Briggs formulas are commonly used for the calculation of the Δh parameter. To simplify the analysis, we only consider the case where the temperature of the pollutant T_s is greater than the ambient air temperature T, which is usually the case. In this case, the value of Δh is given by Formula (2) where F, which denotes the pollutant gas buoyancy, is computed using Formula (3).

$$F = \frac{g}{\pi} \cdot V \cdot (\frac{T_s - T}{T_s}) \tag{3}$$

The Gaussian model considers only one scenario of weather conditions at a time to compute pollution concentrations: Wind direction affects the direction of the pol-

lution plume and variations of the ambient temperature, and the wind velocity affects the concentrations of the pollution plume. Moreover, formula (1) takes into account only pointwise pollution sources and thus cannot be applied to area sources like crossroads and line sources like highways. Multiple extensions have been proposed in the literature to deal with these kinds of pollution sources. In addition, many enhanced systems have been developed based on the Gaussian model to take into account complex meteorological data, the effect of buildings on pollution dispersion, etc.

3.2 Interpolation-Based Prediction Methods

Interpolation methods formulate the estimated concentration $\widehat{\mathscr{Z}_p}$ at a given location $p \in \mathscr{P}$ as a weighted combination of the measured concentrations $\mathscr{Z}_q$, $q \in \mathscr{P} - \{p\}$ [30]. The weights of the measured concentrations $\mathscr{W}_{pq}$ can be evaluated in a deterministic way based on the distance between the location of the measured concentration and the location of the estimated concentration. In this case, which is called the inverse distance weighting interpolation, $\widehat{\mathscr{Z}_p}$ is evaluated using formula (4). The concentration weights can also be evaluated in a stochastic way, the most used method doing so is called kriging.

$$\widehat{\mathscr{Z}_p} = \frac{\sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * \mathscr{Z}_q}{\sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq}} \tag{4}$$

3.3 Regression-Based Prediction Methods

Land-use regression models are a kind of stochastic regression models [31]. The idea behind these models is to evaluate the pollution concentration at a given location based on the concentrations of locations that are similar in terms of land-use parameters such as the elevation and the distance to the closest busy road.

4 Model 1: WSN Deployment for Air Pollution Mapping Based on Predicted Data

In this section, we present an integer programming model of WSN deployment ensuring air pollution mapping. We formulate the constraint of air pollution coverage based on interpolation methods in order to determine the optimal positions of sensors allowing to better estimate pollution concentrations at positions where no sensor is deployed. Our coverage formulation takes into account the sensing drift of sensor

nodes and the impact of weather conditions on air pollution dispersion. We base on the flow problem to formulate the connectivity constraint that ensures that the deployed sensors are able to send pollution data to at least one sink. We provide an evaluation of the ILP model on a data set of the Lyon city while analyzing the coverage and connectivity results.

4.1 Problem Formulation

4.1.1 Overview

We consider as input of our model the map of a given urban area that we call the deployment region. We start by discretizing the deployment region in order to get a set of points $\mathscr{P}$ approximating the urban area at a high-scale ($|\mathscr{P}| = \mathscr{N}$). Our goal is to be able to determine with a high precision the concentration value at each point $p \in P$. We ensure that for each point $p \in P$, either a sensor is deployed or the pollution concentration can be estimated with a high precision based on data gathered by the neighboring deployed sensors.

In general case, the set $\mathscr{P}$ is thus considered as the set of potential positions of WSN nodes. However, in smart cities applications, some restrictions on node positions may apply because of authorization or practical issues. For instance, in order to alleviate the energy constraints, we may place sensors on only lampposts and traffic lights as experimented in [32]. When this is the case, we do not consider as potential positions the points $p \in P$ where sensors cannot be deployed.

We use decision variables x_p (respectively y_p) to specify if a sensor (respectively a sink) is deployed at point p or not. Sensors and sinks may have different costs, thus we denote by c_p^{sensor} (respectively c_p^{sink}) the sensor (respectively the sink) deployment cost at position p. We summarize in Table 2 the notations used in the formulations.

4.1.2 Objective Function

The objective of the ILP model is to minimize the overall financial cost of the resulting network. The cost function to minimize is thus given as follows:

$$\mathscr{F} = \sum_{p \in \mathscr{P}} c_p^{sensor} * x_p + \sum_{p \in \mathscr{P}} c_p^{sink} * y_p \tag{5}$$

4.1.3 Coverage

Basic formulation As claimed before, our idea is to base on interpolation methods in order to ensure that the deployed sensors allow to estimate with a high precision the pollution concentrations at locations where no sensor is deployed. This means

Table 2 Summary of the model notations

Parameters	
$\mathscr{P}$	Set of points approximating the deployment region
$\mathscr{N}$	Number of points approximating the deployment region
$\mathscr{Z}_p$	Reference pollution concentration at point p
$\mathscr{W}_{pq}$	Correlation coefficient between points p and q
$\mathscr{D}$	The correlation distance function
d	Maximum correlation distance
α	Attenuation coefficient of the correlation distance
$\Gamma(p)$	Communication neighborhood of a node deployed at point p
$\mathscr{R}$	Communication range of sensor nodes
$\mathscr{E}_p$	The tolerated estimation error at point p
$\mathscr{M}$	The maximum number of sinks
c_p^{sensor}	The cost of deploying a sensor at point p
c_p^{sink}	The cost of deploying a sink at point p
Variable	
x_p	Define whether a sensor is deployed at point p or not $x_p \in \{0, 1\},\ p \in \mathscr{P}$
y_p	Define whether a sink is deployed at point p or not $y_p \in \{0, 1\},\ p \in \mathscr{P}$
g_{pq}	Flow quantity transmitted from node p to node q $g_{pq} \in \{0, 1, \ldots\},\ p \in \mathscr{P},\ q \in \Gamma(p)$

that we need to have an idea on the dispersion of pollution concentrations in the deployment region in order to be able to formulate the coverage constraint. More exactly, we need to know the variability of pollution concentrations among the set of points P in order to use the formulation of interpolation methods. Fortunately, using numerical atmospheric dispersion models, we can obtain simulated pollution concentrations that may be considered as reference pollution concentrations [33]. This does not mean that these reference concentrations are real, but they reflect the best today's pollution knowledge.

Let $\mathscr{Z}_p$ denote the reference concentration value at point p. Given the set of selected points where sensors will be deployed $\{p\ where\ x_p = 1\}$, we evaluate the estimated pollution concentrations $\widehat{\mathscr{Z}_p}$ at points $\{p\ where\ x_p = 0\}$ based on reference values corresponding to the selected points, i.e., based on $\mathscr{Z}_p\ where\ p \in \{p\ where\ x_p = 1\}$, as follows:

$$\begin{cases} \widehat{\mathscr{Z}_p} = \dfrac{\sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * \mathscr{Z}_q * x_q}{\sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * x_q},\ p \in \mathscr{P}\ \&\ x_p = 0 \\ \displaystyle\sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * x_q > 0,\ p \in \mathscr{P}\ \&\ x_p = 0 \end{cases} \tag{6}$$

The $\widehat{\mathscr{Z}_p}$ expression is formulated based on formula (4) given in Sect. 3.2. We have chosen this formula because the weights $\mathscr{W}_{pq}$ are given in a deterministic way, which allows to integrate them to the ILP deployment model. We ensure that the denominator of $\widehat{\mathscr{Z}_p}$ is never equal to zero using the second part of formula (6). The $\mathscr{W}_{pq}$ parameter is the correlation coefficient between points p and q and is calculated using formula (7) based on the distance between the two points. $\mathscr{D}(p,q)$ is the distance function. α is the attenuation coefficient of the correlation distance; this means that for greater values of α, very low correlation coefficients are assigned to far points. The last parameter of formula (7) is the maximum correlation distance, which defines the range of correlated neighboring points of a given point.

In order to take into account the impact of the urban topography on the dispersion of pollutants, let $\mathscr{D}$ be the shortest distance along the roads network. This allows to assign small correlation values to points that are separated by buildings, even if they are close [34].

$$\mathscr{W}_{pq} = \begin{cases} \frac{1}{\mathscr{D}(p,q)^{\alpha}} & \text{if } q \in Disc(p,d) - \{p\} \\ 0 & \text{if } q \notin Disc(p,d) \end{cases} \tag{7}$$

In order to ensure that the concentration is estimated with high precision at points where no sensor is deployed, we define constraint (8). The $\mathscr{E}_p$ parameter corresponds to the estimation error that is tolerated at point p. The choice of different values of $\mathscr{E}_p$ in function of p allows to assign low tolerated estimation errors to locations that are sensitive to air quality such as hospitals, primary schools.

$$\left|\widehat{\mathscr{Z}_p} - \mathscr{Z}_p\right| \leq \mathscr{E}_p, \quad p \in \mathscr{P} \;\&\; x_p = 0 \tag{8}$$

By replacing $\widehat{\mathscr{Z}_p}$ by its expression given in formula (6), we obtain the coverage constraints (9) and (10). These two constraints should be linearized in order to get an ILP formulation.

$$\left| \frac{\sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * \mathscr{Z}_q * x_q}{\sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * x_q} - \mathscr{Z}_p \right| \leq \mathscr{E}_p,\ p \in \mathscr{P} \;\&\; x_p = 0 \tag{9}$$

$$\sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * x_q > 0,\ p \in \mathscr{P} \;\&\; x_p = 0 \tag{10}$$

Linearization of constraint (9) The first step is to linearize the fraction part; this allows to get constraint (11). Then, we have to ensure that the constraint is relaxed when $x_p = 1$. To do so, notice that the left member of constraint (11) can be bounded as presented in formula (12). Based on this, we add $x_p * \sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * |\mathscr{Z}_q - \mathscr{Z}_p|$ to the right member of constraint (11) to relax it when $x_p = 1$. Hence, we obtain constraint (13). Finally, we have to linearize the absolute value function. Hence, we get the linear form of constraint (9) in constraints (14) and (15).

$$\left| \sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * x_q * (\mathscr{Z}_q - \mathscr{Z}_p) \right| \leq \mathscr{E}_p * \sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * x_q,\ p \in \mathscr{P}, x_p = 0 \quad (11)$$

$$\left| \sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * x_q * (\mathscr{Z}_q - \mathscr{Z}_p) \right| \leq \sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * |\mathscr{Z}_q - \mathscr{Z}_p| \quad (12)$$

$$\left| \sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * x_q * (\mathscr{Z}_q - \mathscr{Z}_p) \right| \leq \mathscr{E}_p * \sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * x_q + x_p * \sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * |\mathscr{Z}_q - \mathscr{Z}_p|,\ p \in \mathscr{P} \quad (13)$$

$$\sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * x_q * (\mathscr{Z}_q - \mathscr{Z}_p) \leq \mathscr{E}_p * \sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * x_q + x_p * \sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * |\mathscr{Z}_q - \mathscr{Z}_p|,\ p \in \mathscr{P} \quad (14)$$

$$\sum_{q \in \mathscr{P} - \{p\}} -\mathscr{W}_{pq} * x_q * (\mathscr{Z}_q - \mathscr{Z}_p) \leq \mathscr{E}_p * \sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * x_q + x_p * \sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * |\mathscr{Z}_q - \mathscr{Z}_p|,\ p \in \mathscr{P} \quad (15)$$

Linearization of constraint (10) The only thing to do to linearize constraint (10) is to relax the constraint when $x_p = 1$. This can be obtained by replacing the right member of the constraint by $-x_p$, which allows to get the constraint (16).

$$\sum_{q \in \mathscr{P} - \{p\}} \mathscr{W}_{pq} * x_q > -x_p,\ p \in \mathscr{P} \quad (16)$$

Taking into account sensing drift Usually, the pollution concentration measured at point q is not equal to the ground truth value $\mathscr{Z}_q$ and depends on the sensing technology and the quality of sensors. This involves a certain drift in pollution measurements. The sensing drift is usually given by two parameters a_q and b_q, which define the measured concentration that is equal to $a_q * \mathscr{Z}_q + b_q$. By introducing parameters a_q and b_q in formula (6), we get in formula (17) the new definition of

the estimated pollution concentration at a given point p depending on the deployed sensors. Using this new definition, we transform the coverage constraints (14) and (15) into constraints (18) and (19), which allows us to include the sensing drift in our coverage model. In this formulation, parameters a_q and b_q are assumed to be constants. When it is not the case, stochastic programming should be used instead of integer programming.

$$\widehat{\mathscr{Z}_p} = \frac{\sum_{q\in\mathscr{P}-\{p\}} \mathscr{W}_{pq} * (a_q * \mathscr{Z}_q + b_q) * x_q}{\sum_{q\in\mathscr{P}-\{p\}} \mathscr{W}_{pq} * x_q} \tag{17}$$

$$\sum_{q\in\mathscr{P}-\{p\}} \mathscr{W}_{pq} * x_q * (a_q \mathscr{Z}_q + b_q - \mathscr{Z}_p) \leq \mathscr{E}_p * \sum_{q\in\mathscr{P}-\{p\}} \mathscr{W}_{pq} * x_q + x_p * \sum_{q\in\mathscr{P}-\{p\}} \mathscr{W}_{pq} * |a_q \mathscr{Z}_q + b_q - \mathscr{Z}_p|, \ p \in \mathscr{P} \tag{18}$$

$$\sum_{q\in\mathscr{P}-\{p\}} -\mathscr{W}_{pq} * x_q * (a_q \mathscr{Z}_q + b_q - \mathscr{Z}_p) \leq \mathscr{E}_p * \sum_{q\in\mathscr{P}-\{p\}} \mathscr{W}_{pq} * x_q + x_p * \sum_{q\in\mathscr{P}-\{p\}} \mathscr{W}_{pq} * |a_q \mathscr{Z}_q + b_q - \mathscr{Z}_p|, \ p \in \mathscr{P} \tag{19}$$

Taking into account weather conditions Air pollution dispersion highly depends on weather conditions such as wind and temperature. For instance, reference pollution concentrations $\mathscr{Z}_p$ can be totally different if there is a change in wind direction. In order to cope with that, we consider multiple snapshots of reference concentrations. The resolution of snapshots may be yearly, monthly, or daily depending on the needed deployment accuracy and the data availability. Let $\mathscr{T}$ be the set of snapshots, and $\mathscr{Z}_p^t$ be the reference pollution concentration at point p in snapshot t. We propose to ensure that constraints (18) and (19) are verified for each snapshot $t \in \mathscr{T}$; hence, we get constraints (20) and (21). This allows us to place sensor nodes while taking into account the different weather scenarios corresponding to each pollution snapshot.

$$\sum_{q\in\mathscr{P}-\{p\}} \mathscr{W}_{pq} * x_q * (a_q \mathscr{Z}_q^t + b_q - \mathscr{Z}_p^t) \leq \mathscr{E}_p * \sum_{q\in\mathscr{P}-\{p\}} \mathscr{W}_{pq} * x_q + x_p * \sum_{q\in\mathscr{P}-\{p\}} \mathscr{W}_{pq} * |a_q \mathscr{Z}_q^t + b_q - \mathscr{Z}_p^t|, \ p \in \mathscr{P}, \ t \in \mathscr{T} \tag{20}$$

$$\sum_{q\in\mathscr{P}-\{p\}} -\mathscr{W}_{pq} * x_q * (a_q \mathscr{Z}_q^t + b_q - \mathscr{Z}_p^t) \leq \mathscr{E}_p * \sum_{q\in\mathscr{P}-\{p\}} \mathscr{W}_{pq} * x_q + x_p * \sum_{q\in\mathscr{P}-\{p\}} \mathscr{W}_{pq} * |a_q \mathscr{Z}_q^t + b_q - \mathscr{Z}_p^t|, \ p \in \mathscr{P}, \ t \in \mathscr{T} \tag{21}$$

4.1.4 Connectivity

We formulate the connectivity constraint as a network flow problem. We consider the same potential positions set $\mathscr{P}$ for sensors and sinks. We first denote by $\Gamma(p),\ p \in \mathscr{P}$, the set of neighbors of a node deployed at the potential position p. This set can be determined using sophisticated path loss models. It can also be determined using the binary disc model, in which case $\Gamma(p) = \{q \in P\ where\ q \in Disc(p, R)\}$ where R is the communication range of sensors. Then, we define the decision variables g_{pq} as the flow quantity transmitted from a node located at potential position p to another node located at potential position q. We suppose that each sensor of the resulting WSN generates a flow unit in the network and verify if these units can be recovered by sinks. The following constraints ensure that the deployed sensors and sinks form a connected wireless sensor network; i.e., each sensor can communicate with at least one sink.

$$\sum_{q\in\Gamma(p)} g_{pq} - \sum_{q\in\Gamma(p)} g_{qp} \geq x_p - (\mathscr{N}+1) * y_p,\ p \in \mathscr{P} \tag{22}$$

$$\sum_{q\in\Gamma(p)} g_{pq} - \sum_{q\in\Gamma(p)} g_{qp} \leq x_p, \quad p \in \mathscr{P} \tag{23}$$

$$\sum_{q\in\Gamma(p)} g_{pq} \leq N * x_p, \quad p \in \mathscr{P} \tag{24}$$

$$\sum_{p\in\mathscr{P}} \sum_{q\in\Gamma(p)} g_{pq} = \sum_{p\in\mathscr{P}} \sum_{q\in\Gamma(p)} g_{qp} \tag{25}$$

$$\sum_{p\in\mathscr{P}} y_p \leq \mathscr{M} \tag{26}$$

Constraints (22) and (23) are designed to ensure that each deployed sensor, i.e., such that $x_p = 1$, generates a flow unit in the network. These constraints are equivalent to the following.

$$\sum_{q\in\Gamma(p)} g_{pq} - \sum_{q\in\Gamma(p)} g_{qp} \begin{cases} =1 & if\ x_p = 1, y_p = 0 \\ =0 & if\ x_p = y_p = 0 \\ \leq 0, \geq -\mathscr{N} & if\ x_p = 1, y_p = 1 \end{cases}$$

The first case corresponds to deployed sensors that should generate, each one of them, a flow unit. The second case, combined with constraint (24), ensures that absent nodes, i.e., $x_p = y_p = 0$, do not participate in the communication. The third case concerns deployed sinks and ensures that each sink cannot receive more than $\mathscr{N}$ units. Constraint (25) means that the overall flow is conservative. The flow sent by deployed sensors has to be received by deployed sinks. Finally, constraint (26) allows to fix the maximum number of sinks $\mathscr{M}$ of the resulting network.

4.1.5 ILP Model

The ILP model allowing for air pollution mapping based on predicted pollution maps is as follows:

$[ILP1]$

$Minimize$ (5)
$Subject\ to.$ (16), (20), (21), (22), (23), (24), (25) *and* (26)

4.2 Simulation Results

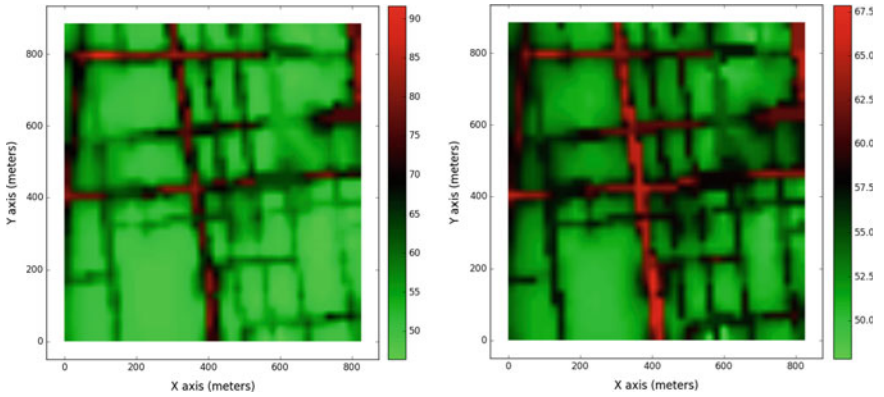
In this section, we present the simulations that we have performed in order to validate our model and analyze its performances. We first present the data set that we have used and the common simulation parameters. Then, we give a proof of concept in order to show how error-bounded deployment is done. Finally, we analyze the results of pollution coverage and network connectivity while studying the compromise between the estimation precision and the deployment cost under different configurations of the correlation distance.

4.2.1 Simulation Dataset

In order to consider the real dispersion of air pollutants in the reference pollution concentrations $\mathscr{Z}_p^t$, we perform our simulations on two pollution snapshots generated by an enhanced atmospheric dispersion simulator called SIRANE [33]. This simulator is designed for urban areas and takes into account the impact of street canyons on pollution concentrations. The data set has been provided by Air-Rhone-Alpes, which is an observatory for air pollution monitoring within the Lyon region of France [35].

We depict the two reference pollution maps in Fig. 1. We focus on nitrogen dioxide (NO_2) monitoring since this pollutant is mainly due to road traffic. We evaluate our ILP model on the La-Part-Dieu district, which is the heart of the Lyon City. Pollution map granularity is around 5 m, and concentrations correspond to the years 2012 and 2013.

We have implemented the ILP formulation using IBM ILOG CPLEX Optimization Studio and used a computer with Intel Xeon E5649 processor under Linux. Simulation parameters are summarized in Table 6. We discretize the deployment region which is of around 0.7 km^2 using a resolution of 50 m; thus, we get 306 discrete points. We consider all these points as potential positions of nodes. We use the same tolerated estimation error $\mathscr{E}_p = \mathscr{E}$ for all the points $p \in \mathscr{P}$. By default, we use the distance along roads for the evaluation of the correlation coefficients and we suppose that sensing is perfect. In addition, we fix the maximum number of sinks to 1 in order to get mono-sink networks (Table 3).



(a) NO_2 concentrations ($\mu g/m^3$) of 2012 (b) NO_2 concentrations ($\mu g/m^3$) of 2013

Fig. 1 Reference NO2 concentrations in Lyon La-Part-Dieu district, average over years 2012 and 2013 (Ref:Air-Rhone-Alpes)

Table 3 Default values of simulation parameters

Parameter	Notation	Value
Number of discrete points	$\mathcal{N}$	306
Maximum correlation distance	d	100 m
Attenuation coefficient of correlation distance	α	2
Communication range of sensor nodes	$\mathcal{R}$	150 m
The tolerated estimation error at point p	$\mathcal{E}_p$	10 μg/m^3
The maximum number of sinks	$\mathcal{M}$	1
The cost of deploying a sensor at point p	c_p^{sensor}	1 unity
The cost of deploying a sink at point p	c_p^{sensor}	10 unities

4.2.2 Proof of Concept

In order to validate our formulation of error-bounded WSN deployment, we ran the ILP model using the two reference pollution maps while considering three values for the tolerated estimation error: 2, 5, and 8 μg/m^3. We depict in Table 4 the resulting optimal cost corresponding to the snapshot of 2012 alone, the snapshot of 2013 alone, and the two snapshots together (using formulations given in Sect. 4.1.3). We notice that snapshot 2012 needs more sensors than snapshot 2013. This is because the range of pollution concentrations is larger in snapshot 2012 as shown in Fig. 1, which involves higher pollution variability and thus more interpolation errors. In addition,

Table 4 Deployment cost (monetary units) depending on snapshots and the tolerated estimation error

Tolerated estimation error	2 $\mu g/m^3$	5 $\mu g/m^3$	8 $\mu g/m^3$
Snapshot 2012 alone	221	146	105
Snapshot 2013 alone	171	67	48
Snapshots 2012 and 2013 together	237	148	105

when trying to satisfy both of the two snapshots, we place at least the sensors that are required by snapshot 2012 since this snapshot is more complicated than the other one.

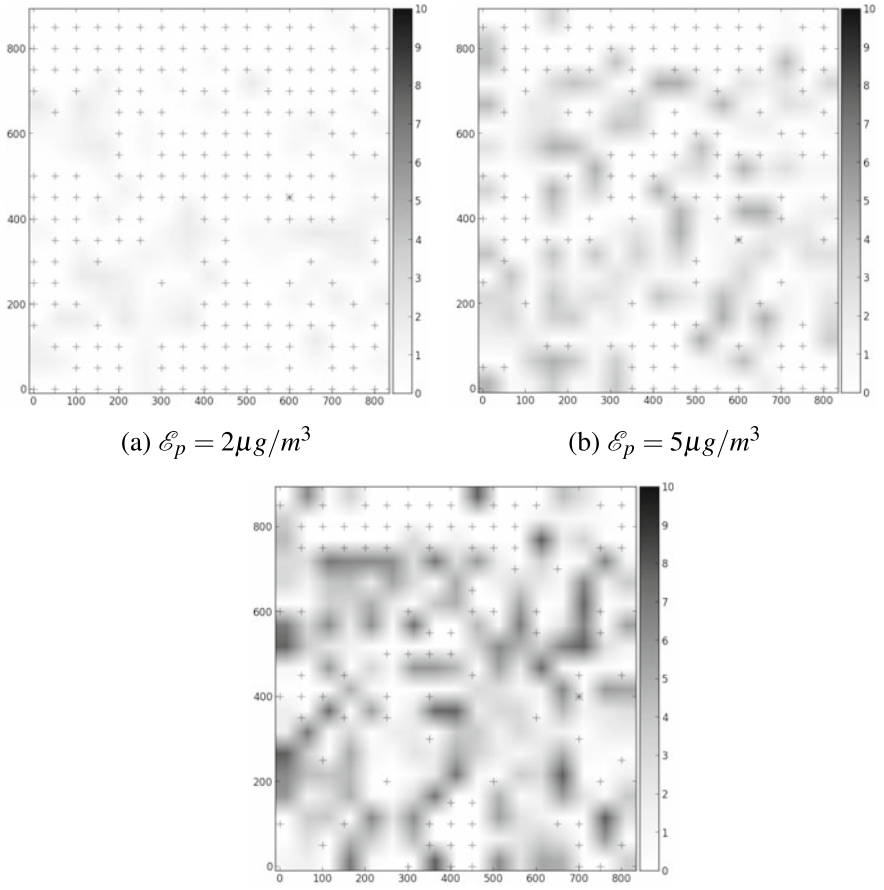
We now depict in Fig. 2 the obtained positions of sensors and sinks when using only snapshot 2012. We evaluate at each point of the map the estimated concentration, and then, we calculate the resulting estimation error, i.e., the difference between the reference concentrations and the estimated concentrations. The obtained errors are also depicted in Fig. 2.

We notice that more sensors are used when the tolerated estimation error decreases. This is expected since better deployment precision needs more sensor nodes. In addition, Fig. 2 shows that the maximum error value is bounded by the tolerated estimation error, which fits with our coverage formulation. Moreover, the obtained nodes form a connected network as formulated in our connectivity constraint. In the next simulation cases, we execute the model only on the snapshot of year 2012.

4.2.3 Evaluation of the Coverage Results

In this simulation scenario, we study the dependency between the deployment precision and the needed number of sensors under different configurations of the correlation distance. Since we are studying the cost of the monitoring precision, we execute only the coverage constraint. We depict in Fig. 3 the optimal deployment cost depending on the tolerated estimation error while considering two different functions of the correlation distance: the Euclidean distance and the distance along roads. We notice in the two curves that fewer sensors are needed when the tolerated estimation error decreases. This is because less tolerated estimation error involves high-precision deployment and thus more nodes. In addition, the Euclidean distance gives less number of sensors, which is explained by the fact that the distance along roads is more realistic and hence involves more nodes to better estimate pollution concentrations.

We now investigate the impact of the correlation distance on the number of sensors that are needed to cover a point where no sensor is deployed. We consider different configurations of the maximum correlation distance d and the attenuation coefficient of the correlation distance α. We depict in Fig. 4 the average number of sensors that are



(a) $\mathscr{E}_p = 2\mu g/m^3$ (b) $\mathscr{E}_p = 5\mu g/m^3$

(c) $\mathscr{E}_p = 8\mu g/m^3$

Fig. 2 Deployments with increasing estimation errors ($\mu g/m^3$) for snapshot 2012. Sensors (respectively sinks) are depicted using "plus signs" (respectively stars)

deployed within the maximum correlation distance d of each point where no sensor is deployed. We notice that fewer sensors are used when considering greater values of the attenuation coefficient α and less values of the maximum correlation distance d. To explain this, we recall that smaller values of the d parameter allow to consider less points in the interpolation formula, and higher values of the α parameter allow to assign smaller values of correlation coefficients to the far sensors. This means that with smaller values of d and higher values of α, the interpolation is done with fewer sensors, which fits with the results depicted in Fig. 4.

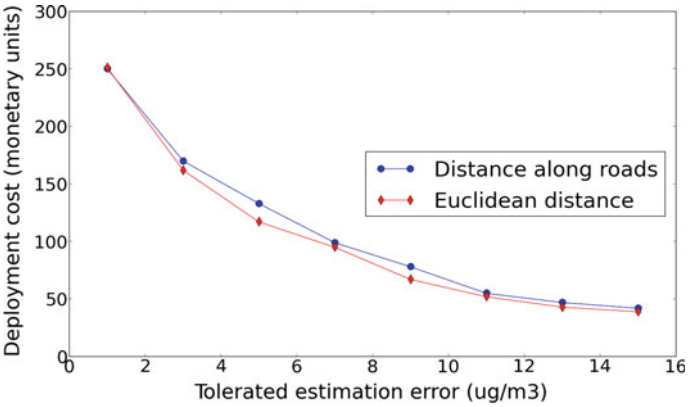
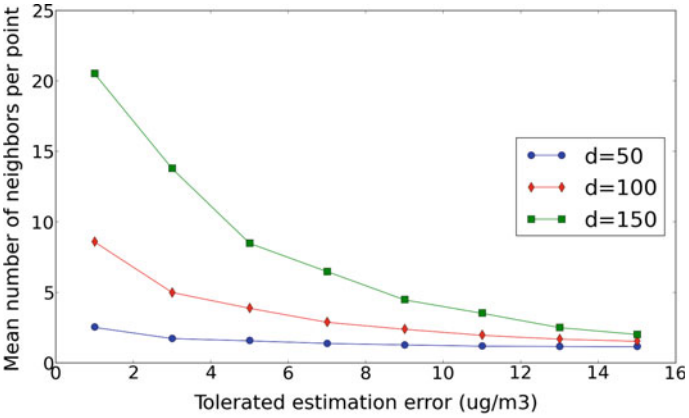


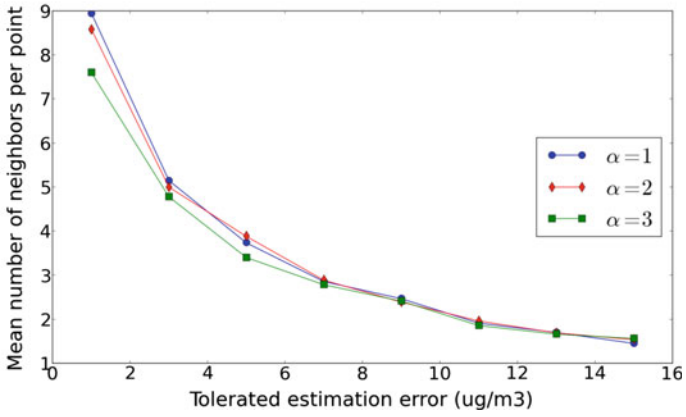
Fig. 3 Optimal coverage cost versus tolerated estimation error

4.2.4 Evaluation of the Network Connectivity

In this simulation case, we analyze the deployment cost that is due to the connectivity constraint, which involves the deployment of sink and relay nodes. We consider three possible values of the communication range of sensor nodes: 80 m for short-range communications like ZigBee, 150 m for medium-range communications like WiFi, and 1000 m for long-range communications. We evaluate the number of nodes depending on the tolerated estimation error and depict the results in Fig. 5. Obviously, the longer the communication range, the less the number of sensors is. However, the tolerated estimation error has a considerable impact on the connectivity of the network. On the one hand, the medium- and long-range communications involve nearly the same number of nodes. For instance, when estimation errors are less than 10 $\mu g/m^3$, the medium-range communications need at most only two more nodes than the high-range communications. This is explained by the fact that small tolerated errors imply a very high density of the network so that sensors are placed almost in all positions. On the other hand, short-range communications are very costly and need almost 70% more nodes than the long-range communications when the estimation errors reach 15 $\mu g/m^3$. This is because the distance between sensor nodes that are responsible for coverage is very important when high estimation errors are tolerated, which causes the need of too much relay nodes if the communication range is very short.



(a) Impact of the maximum correlation distance



(b) Impact of the attenuation coefficient of correlation distance

Fig. 4 Impact of correlation distance parameters on the mean number of neighbors per point

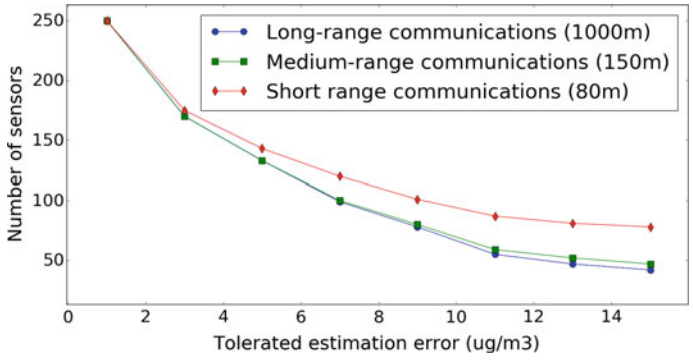


Fig. 5 Optimal number of sensors depending on the communication range

5 Model 2: WSN Deployment for Air Pollution Detection Based on Predicted Data

In this section, we present an integer programming model of WSN deployment ensuring air pollution detection based on spatial analysis of air pollution predicted maps. We consider as input of the model a set of air pollution zones where sensors have to be deployed. This set of zones is determined using ZSCAN, an extension of DBSCAN which is a well-known and widely used spatial data clustering algorithm [36]. The ILP model that we present in this section allows to perform a minimum-cost deployment of sensors while ensuring pollution coverage and network connectivity in a joint way. We evaluate the ILP model on a data set of the Paris city and analyze the impact of some parameters on the deployment results.

5.1 Problem Formulation

5.1.1 Overview

In order to deploy sensors efficiently in a given city, the ILP model requires as a primary input, air pollution predicted maps. Given a pollutant to monitor, this consists of estimated values of the pollutant concentrations in the whole city for different time instants. As explained in Sect. 3, these concentrations can be estimated using atmospheric dispersion modeling based on the locations of pollution sources and meteorological data. They can be also obtained using interpolation algorithms based on some measurements established by a set of monitoring stations.

In the following, we denote by $\mathscr{T}$ the set of time instants when pollution is estimated, and $\mathscr{I}$ the set of spatial points representing the city. The second set is defined by applying a high-resolution discretization process. For each time instant $t \in \mathscr{T}$ and spatial point $i \in \mathscr{I}$, let $C_{t,i}$ denote the estimated or measured pollution concentration. In addition to air pollution estimated concentrations, the model also requires data on sensor potential positions, and this corresponds to positions where sensors can be deployed. We denote in what follows this set by $\mathscr{P}$.

Before the execution of the ILP model, a spatial clustering algorithm is applied to the air pollution maps in order to determine pollution zones that are due to the same pollutant sources. The ILP model allows to define a mono-sink connected wireless sensor network that covers all the pollution zones $z \in \mathscr{Z}$ while minimizing the deployment cost. Pollution monitoring is ensured by deploying at least one sensor in each pollution zone to ensure the coverage constraint. Once the deployment positions of the connected sensors are determined, the sink location can be defined among sensor positions in order to optimize energy consumption, end-to-end transmission delay, etc. The sets used in the ILP model are summarized in Table 5.

Table 5 Summary of the model sets

Notation	Description
$\mathscr{T}$	Set of time instants of pollution estimations
$\mathscr{I}$	Set of discrete points of pollution estimations
$\mathscr{P}$	Set of potential positions of sensors and sinks
$\mathscr{Z}$	Set of pollution zones

5.1.2 Identification of Pollution Zones

We present hereafter ZSCAN, a clustering algorithm which is applied to the estimated concentrations in order to group points that belong to the same pollution zone.

Algorithm 1 ZSCAN

Inputs: $\mathscr{T}, \mathscr{I}, \{C_{t,i}; t \in \mathscr{T}, i \in \mathscr{I}\}$
Output: $\mathscr{Z}$
$\mathscr{Z} \leftarrow \emptyset$
for $t \in \mathscr{T}$ **do**
  Mark all the points in $\mathscr{I}$ as unvisited
  repeat
    Let i be the unvisited point having the highest concentration $C_{t,i}$
    $z \leftarrow construct(i, t)$
    Mark all the points in z as visited
    $z \leftarrow filter(z, \Delta C)$
    $z \leftarrow points\_to\_polygon(z)$
    $\mathscr{Z} \leftarrow \mathscr{Z} \cup \{z\}$
  until all the points in $\mathscr{I}$ are visited

As presented in Algorithm 1, ZSCAN identifies all the pollution zones occurring in each time instant. To this end, pollution peaks, i.e., points having the highest pollution concentration, are first identified. A pollution zone is created every time a peak is identified using the *construct* function, which starts by adding all the neighbors of the pollution peak i to the zone under construction. The neighborhood of a point in the map is defined as the set of closer and unvisited points whose pollution concentration estimated in t is less. The neighbors of each chosen point are then added to the current zone. This process stops when it arrives at a point whose neighborhood set is empty, meaning that its neighbors have higher pollution concentration values. Once the current pollution zone is completely identified, a filtering function is applied to keep only points where pollution concentration is sufficiently closer the peak value, i.e., points where pollution concentration difference with the peak value is less than a threshold that we denote by ΔC. This increases the chance that a sensor deployed in the detected zone is able to monitor the corresponding pollution sources.

At the end, a geometrical form is given to the found zone by applying the function points_to_polygon.

The main difference between DBSCAN and ZSCAN is that this latter visits points in an ordered way, starting by pollution peaks, and then expands the peak neighborhood to detect pollution zones.

5.1.3 ILP Deployment Model

Let $G = \{V, A\}$ be a flow-oriented graph where vertices set V corresponds to the pollution zones and the sensors potential positions, i.e., $\mathscr{Z} \cup \mathscr{P}$. Note that each zone is considered as a single vertex. Let $A(i)$ denote the neighborhood of $i \in V$. We define a first set of arcs from each pollution zone $z \in \mathscr{Z}$ to sensor potential positions p that are within its region, i.e., $p \in z$. A second set of arcs is defined from each sensor potential position $p \in \mathscr{P}$ to positions which are in its communication range that we denote by $\Gamma(p)$.

The idea behind the formulation is that each pollution zone inserts one flow unit in the network through the first set of arcs. For a given selected positions of sensor nodes, the latter ensures network coverage and connectivity if and only if the received units from pollution zones can be forwarded by these nodes through the second set of arcs so that a chosen pollution zone can recover all the generated units (which ensures the connectivity of the defined graph). This particular zone does not generate units but recovers all of them from a sensor that is placed within its region. With these considerations, the selected sensor positions ensure jointly coverage and connectivity if the recovering pollution zone gets all the flow units generated by the other pollution zones and forwarded by the selected sensors. Indeed, flow passing in the first set of arcs guaranties coverage, and connectivity is verified due to flow forwarded by only selected positions through the second set of arcs. In what follows, we choose the first pollution zone $\mathscr{Z}^0$ to be the one that recovers flow units, meaning that the other zones generate, each one, a unique flow unit.

We use binary decision variables x_p to specify if a sensor should be placed at a position p or not. Sensors cost may depend on their positions, thus we denote by $cost_p$ the cost of deploying a sensor at a position p. We also define the positive integer decision variables f_{ij} as the flow quantity transmitted from i to j. The flow domain is set to $\{0, |\mathscr{Z}| - 1\}$ for $j = \mathscr{Z}^0$ in order to ensure that $\mathscr{Z}^0$ recovers the units from only one sensor.

The proposed model **ILP 2** minimizes the overall deployment cost as formulated in the objective function. Constraints (28) ensure that each pollution zone except the first one generates exactly a flow unit. Sensors are flow conservative thanks to constraints (30). Constraints (31) ensure that sensors that are not selected ($x_p = 0$) do not participate in communication. This means that generated flow units will be transmitted by only present sensors. Finally, the first pollution zone receives all the generated units, thanks to constraint (29).

$$[\text{ILP 2}]\ Min \sum_{p \in \mathscr{P}} cost_p * x_p \tag{27}$$

$$S.T.$$

$$\sum_{p \in z} f_{zp} = 1, \quad z \in \mathscr{Z} - \{\mathscr{Z}^0\} \tag{28}$$

$$\sum_{p \in \mathscr{Z}^0} f_{p\mathscr{Z}^0} = |\mathscr{Z}| - 1 \tag{29}$$

$$\sum_{q \in A(p)} f_{pq} - \sum_{q \in A(p)} f_{qp} = 0, \quad p \in \mathscr{P} \tag{30}$$

$$\sum_{q \in A(p)} f_{pq} <= (|\mathscr{Z}| - 1) * x_p, \quad p \in \mathscr{P} \tag{31}$$

After the execution of the ILP model, one of the selected sensor positions can be designated as the sink position since the resulting sensor positions form a connected network. As we are investigating the use of urban facilities in this work, i.e., energy constraints are alleviated, we will consider in the simulation part the sensor-to-sink delay as the main metric for the location of the sink position. One way to do that is to first determine shortest paths between all the pairs of the sensor positions using the delay metric. Then, an optimization model similar to the works presented in [15, 16] can be used in order to find an optimal sink location that minimizes the maximum delay in the network.

5.2 Simulation Results

In this section, we present the simulations that we have performed in order to validate our model and analyze its performances. We first present the data set that we have used and the common simulation parameters. Then, we give a proof of concept showing the use of the ZSCAN algorithm and validating the ILP deployment model. Finally, we analyze the impact of some parameters on the coverage and connectivity results.

5.2.1 Simulation Dataset

We evaluate the ILP model on a pollution data set of the Paris city, while we focus on monitoring NO_2 pollutant particles (Nitrogen dioxide). The pollution data set was provided by Airparif, a French air quality monitoring association. We based on pollution data measured by 22 monitoring stations to estimate pollutant concentrations in the whole city. Because pollution achieved maximum values in Paris in

March 2014 [37], we decided to base on pollution estimation for some periods of this month where pollution concentrations were high. Overall, we constructed ten snapshots of pollution concentrations. We used the kriging interpolation method to estimate the pollution values with a map resolution of 100 m. A set of 21201 spatial data records was then obtained for each snapshot. In addition to pollution data, we used lamppost locations as sensors potential positions $\mathscr{P}$. The lampposts data set was provided by the open data service of the Paris city. We summarize in Table 6 the common values of simulation parameters.

5.2.2 Proof of Concept

First, we execute the ZSCAN algorithm to identify pollution zones where sensors will be deployed. Pollution peaks are detected and then expanded to form these zones. Points in each obtained pollution zone have as maximum concentration difference with the pollution peak of the zone 5 $\mu g/m^3$, and this corresponds to the value of the parameter ΔC as mentioned in Table 6. A number of pollution zones are extracted from each time snapshot. The execution of the spatial clustering algorithm identified 29 pollution zones that occurred in March 2014; these zones are depicted in Fig. 6. We notice that some zones occur in different snapshots with a little bit different shape. This is because these zones correspond usually to the same pollutant sources; the different shapes are due to the evolution of weather conditions.

In order to alleviate the execution of the ILP model, we propose to group pollution zones that share intersections into a region site where a mono-sink sensor network will be deployed. In this simulation case, four sites were identified and are illustrated in Fig. 6.

We now execute the **ILP 2** optimization model in order to find sensor optimal positions based on the generated pollution zones. The ILP model is executed on each site to locate sensors ensuring coverage and connectivity. The results are depicted in Fig. 6. The latter shows that sensors are placed in intersections in order to minimize the financial cost. In addition to sensors placed to ensure the coverage of pollution zones, Fig. 6 shows that some sensors are deployed to serve as relay nodes and hence ensure the network connectivity.

Table 6 Default values of simulation parameters

Parameter	value
Map discretization resolution (for set $\mathscr{I}$)	100 m
Number of time instants (set $\mathscr{T}$)	10
ΔC (used in ZSCAN)	5 $\mu g/m^3$
Sensors communication range (used for $\Gamma(p)$, $p \in \mathscr{P}$)	100 m
Sensors cost ($cost_p$, $p \in \mathscr{P}$)	1 unity (constant)

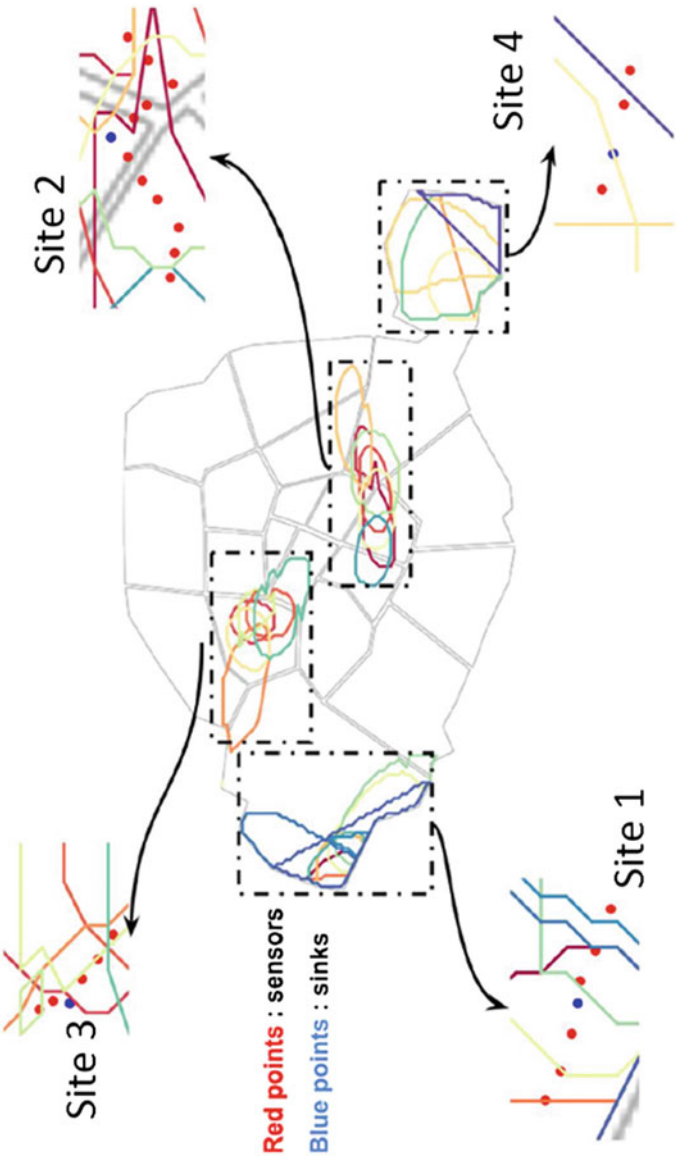


Fig. 6 Proof of concept

As mentioned in the previous section, we base on the works of [15, 16] in order to find the optimal positions of sinks while minimizing the maximum sensor-to-sink delay. We consider in our tests that the delay is linearly proportional to the hop count to the sink. Results are depicted in the same Fig. 6. We notice that sinks are placed in the center of each sub-network, which minimizes the maximum hop count and thus the maximum sensor-to-sink communication delay.

5.2.3 Evaluation of the Coverage Results

In this section, we assess the coverage of the model ILP 2 while showing a comparison to the literature generic formulation presented in Sect. 2 (i.e., coverage and connectivity are modeled independently, coverage is formulated by analogy to the set covering problem using a basic detection model, and connectivity formulation is based on the flow concept).

The coverage formulation of ILP 2 takes into account the nature of the phenomenon based on spatial analysis of pollution data. This is not the case of the generic formulation given in the literature, which assumes that a sensor is able to detect pollutants within a detection range; i.e., pollution is homogeneous within the detection range of a sensor. Even though this assumption is unrealistic since sensors can only detect pollutants that come into their contact, we compare in this simulation scenario the coverage cost given by our model and the generic formulation.

We depict in Fig. 7 the coverage deployment cost of sites 2 and 3 obtained using the generic formulation while considering different values of the detection range compared to our model. Figure 7 shows that our model is at least 5 times better than the generic approach when the detection range is less than or equal to 500 m. The

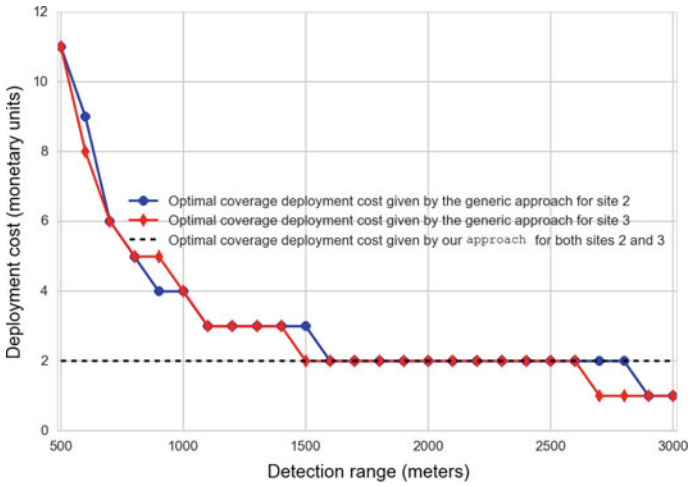


Fig. 7 Optimal coverage deployment cost given by the generic formulation depending on the detection range of sensors

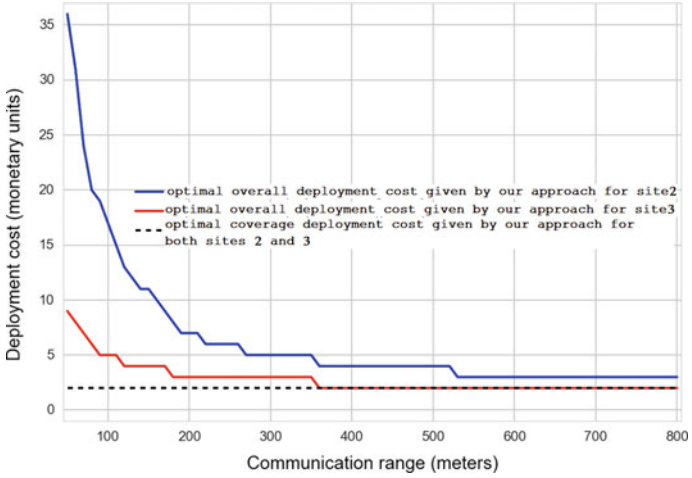


Fig. 8 Optimal deployment cost given by our model depending on the communication range of nodes

coverage cost computed by the generic approach decreases when the detection range increases. However, our model remains better since pollution cannot be homogeneous, as assumed in the generic approach, especially within very large detection ranges.

5.2.4 Evaluation of the Network Connectivity

We now analyze the deployment cost given by the model ILP 2 for sites 2 and 3 while considering different values of the communication range of nodes. We plot in Fig. 8 the obtained results. We recall that the optimum value of deployment cost when connectivity constraint is not taken into account is equal to 2 as shown in Fig. 8. We notice that the larger the communication range, the smaller the deployment cost is. This is expected since using a larger communication range allows to minimize the number of nodes used to connect sensors which are positioned within pollution zones. Figure 8 also shows that when the communication range increases significantly, the overall deployment cost tends to the coverage deployment cost.

6 Model 3: WSN Deployment for Air Pollution Detection Based on Emission Inventory

In this section, we present an ILP optimization model for the deployment of WSN for air pollution detection. Based on the pollution dispersion modeling applied to pollution emission inventory and the ILP related works, we first present pollution

coverage and network connectivity independently. Then, we show how both coverage and connectivity can be formulated in a joint way using the flow concept. The ILP model takes into account the probabilistic sensing of pollution sensors and is designed to handle multiple scenarios of weather conditions. We apply the model on a data set of the London city in order to assess the impact of the model inputs on the deployment results.

6.1 Problem Formulation

6.1.1 Overview

We base our coverage formulation on a pollution dispersion model. For the sake of clarity, we use the Gaussian dispersion model presented in Sect. 3.1 in order to define the inputs of the ILP deployment model. However, these inputs can be also provided by other dispersion models, which may take into account the impact of buildings and urban structures. Pollution sources include industrial sources as well as traffic sources such as highways and crossroads. Table 7 depicts the main notations used in the integer programming model.

In the following, we consider a set of a predefined potential positions, denoted $\mathscr{P}$, which is obtained by discretizing the deployment field while considering only the allowed positions. In free space environments without deployment restrictions, that would be a regular grid. We denote $\mathscr{N} = |\mathscr{P}|$ the number of potential positions. The locations of pollution sources, e.g., factories, sewage treatment plants, crossroads, highways, are denoted $\mathscr{I}$. $\mathscr{M}$ denotes the number of pollution sources. Let the binary decision variables x_p, resp. y_p, define if a sensor, resp. a sink, is placed at position p.

We consider that sinks are equipped with pollution sensors. They are also connected to a backbone network. Deploying a sink is therefore more expensive than a regular sensor node. The cost of deploying a sensor, resp. a sink, at position p is denoted c_p^{sensor}, resp. c_p^{sink}.

6.1.2 Objective Function

The optimization model minimizes the sensors and sinks overall deployment cost. Thus, the objective function is defined as follows:

$$\mathscr{F} = \sum_{p \in \mathscr{P}} c_p^{sensor} \cdot x_p + \sum_{p \in \mathscr{P}} c_p^{sink} \cdot y_p \tag{32}$$

Since a sink embeds sensing capabilities, a sink and a sensor cannot be deployed at the same potential position p as formulated in constraint (33).

Table 7 Main notations used in the deployment model

Sets	
$\mathscr{P}$	Set of potential positions of sensors and sinks
$\mathscr{N}$	Number of sensors and sinks potential positions
$\mathscr{I}$	Set of pollution sources
$\mathscr{M}$	Number of pollution sources
$\mathscr{S}$	Set of weather scenarios
Parameters	
$\mathscr{Z}_i^s$	The pollution zone formed by source i under scenario s
$\mathscr{B}_{ip}^s$	Define whether the position p belongs to the zone Z_i^s or not
$\Gamma(p)$	The neighborhood of the potential position p
c_p^{sensor}	The cost of deploying a sensor at position p
c_p^{sink}	The cost of deploying a sink at position p
β	Minimum coverage probability to ensure for each zone
$\mathscr{W}_{ip}^s$	The probability of detecting the zone Z_i^s at position p
δ	Percentage of scenarios that have to be taken into account
α_s	Probability that scenario s is realized
C_0	Pollutant concentration threshold
Variables	
x_p	Define whether a sensor is deployed at position p or not $x_p \in \{0, 1\},\ p \in \mathscr{P}$
y_p	Define whether a sink is deployed at position p or not $y_p \in \{0, 1\},\ p \in \mathscr{P}$
t_i^s	Define whether the zone $\mathscr{Z}_i^s$ is covered or not $t_i^s \in \{0, 1\},\ i \in \mathscr{I},\ s \in \mathscr{S}$
g_{pq}	Flow quantity transmitted from node p to node q $g_{pq} \in \{0, 1, \ldots\},\ p \in \mathscr{P},\ q \in \Gamma(p)$
f_{ip}^s	Flow quantity transmitted from zone $\mathscr{Z}_i^s$ to node p $f_{ip}^s \in \{0, 1\},\ i \in \mathscr{I},\ s \in \mathscr{S},\ p \in \mathscr{Z}_i^s$

$$x_p + y_p \leq 1, \quad p \in \mathscr{P} \tag{33}$$

6.1.3 Coverage

The coverage constraints rely on the modeling of the atmospheric dispersion. We assume that pollution sources release pollutants independently and may have simultaneous release. Our formulation ensures the coverage of threshold crossings in all cases. As explained in Sect. 3.1, pollution concentrations vary depending on weather conditions. Hence, we consider a set of possible weather scenarios $\mathscr{S}$ that can be obtained based on statistical data or weather forecast. A scenario corresponds to a tuple of ambient temperature T, wind velocity V_w, and wind direction

D_w: $s = (T^s, V_w^s, D_w^s)$. Each scenario s has probability α_s to happen. We assume that $\mathscr{S}$ is a partition of the space of weather conditions, i.e., $\sum_{s \in \mathscr{S}} \alpha_s = 1$ and $s1 \cap s2 = \emptyset$, $\forall s1, s2 \in S$. Here, $s1 \cap s2 = \emptyset$ if the weather scenario $s1$ occurs independently from $s2$.

Using an atmospheric dispersion model, we determine the set of generated pollution zones. Each zone $\mathscr{Z}_i^s$ corresponds to the geographical area, i.e., set of positions, where the pollution threshold is crossed when the pollution source i is releasing pollutants under the weather scenario s.

Let the binary parameter $\mathscr{B}_{ip}^s$ denote whether a position p belongs to $\mathscr{Z}_i^s$ or not. A pollution zone $\mathscr{Z}_i^s$ is therefore the set $\{p \in \mathscr{P} \; where \; \mathscr{B}_{ip}^s = 1\}$. When using the pointwise Gaussian dispersion model, the value of $\mathscr{B}_{ip}^s$ is calculated using Formula (34) where σ_y, σ_z, Q, and H are the parameters presented in Sect. 3, $p = (x, y, z)$ and C_0 is the threshold of pollutant concentration above which a point is considered as polluted.

$$\mathscr{B}_{ip}^s = \begin{cases} 1 \text{ if } \frac{Q}{2\pi V_w^s \sigma_y \sigma_z} e^{-\frac{y^2}{2\sigma_y}} (e^{-\frac{(z-H)^2}{2\sigma_y}} + e^{-\frac{(z+H)^2}{2\sigma_y}}) \geq C_0 \\ 0 \text{ otherwise} \end{cases} \tag{34}$$

A sensor exposed to a given pollutant will detect its concentration with a probability depending on the sensing accuracy. We denote $\mathscr{W}_{ip}^s \in]0, 1[$ the probability of detecting the pollution source i under the weather scenario s at position p, $p \in Z_i^s$. The $\mathscr{W}_{ip}^s$ parameters are mainly due to the technical characteristics of pollution sensors and are not related to the dispersion model.

Once the pollution zones $\mathscr{Z}_i^s$ are identified and the probability parameters $\mathscr{W}_{ip}^s$ are computed, we formulate the coverage of each pollution source i under each weather scenario s with a probability β in constraint (35).

$$\prod_{p \in \mathscr{Z}_i^s} (1 - \mathscr{W}_{ip}^s \cdot (x_p + y_p)) \leq (1 - \beta), \quad i \in \mathscr{I}, \; s \in \mathscr{S} \tag{35}$$

When a sensor or a sink is placed at position p, i.e., $x_p + y_p = 1$, $1 - \mathscr{W}_{ip}^s \cdot (x_p + y_p)$ is then equal to $1 - \mathscr{W}_{ip}^s$, the probability that the node deployed at p do not cover the pollution zone Z_i^s at position p. Assuming that the detection events are independent among all potential positions, constraint (35) ensures therefore that each zone Z_i^s is covered with a probability $\beta \in]0, 1[$.

Constraint (35) ensures that each pollution source is covered with a probability β under each scenario s. One could want to relax this constraint and ask only for the coverage of each pollution source under δ percent of weather scenarios, with a β probability for each scenario. For that, we introduce the binary variable t_i^s that define whether source i is covered during weather scenario s. Therefore, $t_i^s = 1$ if a sufficient number of sensors is placed in the pollution zone Z_i^s. The percentage of weather conditions where i can be detected is the sum of the probabilities that a scenario in which i is detected happens. As a result, the coverage formulation of the partial coverage case is given by the constraints (36) and (37).

$$\prod_{p \in \mathscr{Z}_i^s} (1 - \mathscr{W}_{ip}^s \cdot (x_p + y_p)) \leq (1 - \beta \cdot t_i^s), \quad i \in \mathscr{I}, s \in \mathscr{S} \tag{36}$$

$$\sum_{s \in \mathscr{S}} t_i^s \cdot \alpha_s \geq \delta, \quad i \in \mathscr{I} \tag{37}$$

Constraint (36) should be linearized in order to get an ILP formulation. We first apply the log function.

$$\log \prod_{p \in \mathscr{Z}_i^s} (1 - \mathscr{W}_{ip}^s \cdot (x_p + y_p)) \leq \log(1 - \beta \cdot t_i^s), \ i \in \mathscr{I}, s \in \mathscr{S} \tag{38}$$

$$\sum_{p \in \mathscr{Z}_i^s} \log(1 - \mathscr{W}_{ip}^s \cdot (x_p + y_p)) \leq \log(1 - \beta \cdot t_i^s), \ i \in \mathscr{I}, s \in \mathscr{S} \tag{39}$$

Since $x_p + y_p$ and t_i^s are binary, the log can be rewritten as follows and constraint (40) is linear.

$$\sum_{p \in \mathscr{Z}_i^s} (x_p + y_p) \cdot \log(1 - \mathscr{W}_{ip}^s) \leq t_i^s \cdot \log(1 - \beta), \ i \in \mathscr{I}, s \in \mathscr{S} \tag{40}$$

6.1.4 Connectivity

As in [12, 38] and [39], we formulate in this first model the connectivity constraint as a network flow problem. In contrast to these works, we consider the same potential positions set $\mathscr{P}$ for sensors and sinks and we do not assume that potential positions of sinks are known or different from those of sensors. We first denote by $\Gamma(p), \ p \in \mathscr{P}$ the set of neighbors of a node deployed at the potential position p. This set can be computed using any adequate propagation models. Then, we define the decision variables g_{pq} as the flow quantity transmitted from a node located at potential position p to another node located at potential position q. We suppose that each sensor of the resulting WSN generates a flow unit in the network and verify if these units can be recovered by sinks. The following constraints ensure that deployed sensors and sinks form a connected wireless sensor network; i.e., each sensor can communicate with at least one sink.

$$\sum_{q \in \Gamma(p)} g_{pq} - \sum_{q \in \Gamma(p)} g_{qp} \geq x_p - \mathscr{N} \cdot y_p, \quad p \in \mathscr{P} \tag{41}$$

$$\sum_{q \in \Gamma(p)} g_{pq} - \sum_{q \in \Gamma(p)} g_{qp} \leq x_p, \quad p \in \mathscr{P} \tag{42}$$

$$\sum_{q \in \Gamma(p)} g_{pq} \leq \mathscr{N} \cdot x_p, \quad p \in \mathscr{P} \tag{43}$$

$$\sum_{p\in\mathscr{P}}\sum_{q\in\Gamma(p)} g_{pq} = \sum_{p\in\mathscr{P}}\sum_{q\in\Gamma(p)} g_{qp} \tag{44}$$

Constraints (41) and (42) are designed to ensure that each deployed sensor, i.e., such that $x_p = 1$, generates a flow unit in the network. These constraints are equivalent to the following.

$$\sum_{q\in\Gamma(p)} g_{pq} - \sum_{q\in\Gamma(p)} g_{qp} \begin{cases} =1 & if\ x_p = 1, y_p = 0 \\ =0 & if\ x_p = y_p = 0 \\ \leq 0, \geq -\mathscr{N} & if\ x_p = 0, y_p = 1 \end{cases}$$

The first case corresponds to deployed sensors that should generate, each one of them, a flow unit. The second case, combined with constraint (43), ensures that absent nodes, i.e., $x_p = y_p = 0$, do not participate in the communication. The third case concerns deployed sinks and ensures that each sink cannot receive more than $\mathscr{N}$ units. The case $x_p = y_p = 1$ is not possible because of constraint (33). Constraint (43) ensures also that deployed sinks cannot transmit flow units and only act as receivers. Constraint (44) means that the overall flow is conservative. The flow sent by deployed sensors has to be received by deployed sinks.

6.1.5 ILP Model

Now, we propose to combine coverage and connectivity constraints defined in the two previous sections using only the flow concept. By considering pollution sources as a part of the network, we obtain a homogeneous coverage/connectivity formulation as a network flow problem. Each pollution source i should transmit some flow units to potential nodes p which are located within the pollution zone corresponding to each weather scenario s, i.e., $p \in \mathscr{Z}_i^s$. In addition, sensors are flow conservative, and the sinks receive the flow units generated by pollution sources. Therefore, the definition of the joint coverage/connectivity is to ensure that sinks will be informed each time that a threshold crossing occurs. In this regard, a sensor has to receive at most one unit from a given pollution zone. We hence define the binary decision variable f_{ip}^s as the flow quantity from the pollution source i to the potential node p in the case of weather scenario s. The following constraints ensure coverage and connectivity for air pollution detection in a joint way.

$$\sum_{s\in\mathscr{S}} t_i^s \cdot \alpha_s \geq \delta, \quad i \in \mathscr{I} \tag{45}$$

$$\sum_{p\in\mathscr{Z}_i^s} f_{ip}^s \cdot \log(1 - \mathscr{W}_{ip}^s) \leq t_i^s \cdot \log(1-\beta), i \in \mathscr{I}, s \in \mathscr{S} \tag{46}$$

$$\sum_{i\in\mathscr{I},s\in\mathscr{S}:p\in\mathscr{Z}_i^s} f_{ip}^s + \sum_{q\in\Gamma(p)} g_{qp} - g_{pq} \leq \mathscr{N}\mathscr{M}|\mathscr{S}|y_p, \quad p \in \mathscr{P} \tag{47}$$

$$\sum_{i\in\mathscr{I},s\in\mathscr{S}:p\in\mathscr{Z}_i^s} f_{ip}^s + \sum_{q\in\Gamma(p)} g_{qp} - g_{pq} \geq 0, \quad p \in \mathscr{P} \tag{48}$$

$$\sum_{q\in\Gamma(p)} g_{pq} \leq \mathscr{N}\mathscr{M}|\mathscr{S}|x_p, \quad p \in \mathscr{P} \tag{49}$$

$$f_{ip}^s \leq x_p + y_p, \quad p \in \mathscr{P}, i \in \mathscr{I}, s \in \mathscr{S} \tag{50}$$

Coverage is formulated in constraints (45) and (46). Constraint (45) ensures coverage of each pollution source under a δ percentage of weather scenarios. Constraint (46) is derived from constraint (40) and ensures that each pollution source i generates a sufficient number of flow units in the network. Constraint (50) enforces that all the flow units are received by deployed nodes. Thanks to constraints (47) and (48), when a sensor is deployed on point p (case $y_p = 0$ and $x_p = 1$), we ensure that the inflow of sensor p is equal to its outflow; i.e., the flow is conservative on deployed sensors. In addition, constraints (47) and (48) also ensure that the sinks are allowed to gather all the flow units that are generated in the network (case $y_p = 1$ and $x_p = 0$). Constraints (49) and (50) combined with constraints (47) and (48) ensure that absent nodes do not participate in the communication. As a result, the deployed sensors have to send the flow units gathered from pollution sources to the sinks in order to get the connectivity constraints verified.

Finally, the ILP optimization model can be written as follows:

$[ILP\ 3]$
$Minimize$ (32)
$Subjectto.$ (33), (45), (46), (47), (48), (49) *and* (50)

6.2 Simulation Results

In this section, we present the simulations that we have performed to evaluate the ILP 3 deployment model. We first present the data set of Greater London that we used in simulation. Then, as a proof of concept, we apply the model to the London Borough of Camden. Finally, we study the impact of the sink/sensor cost ratio, nodes height, pollution sources density, the probabilistic sensing of sensors, and weather conditions. This study allows us to derive engineering insights for effective deployments of air pollution sensors in an urban environment.

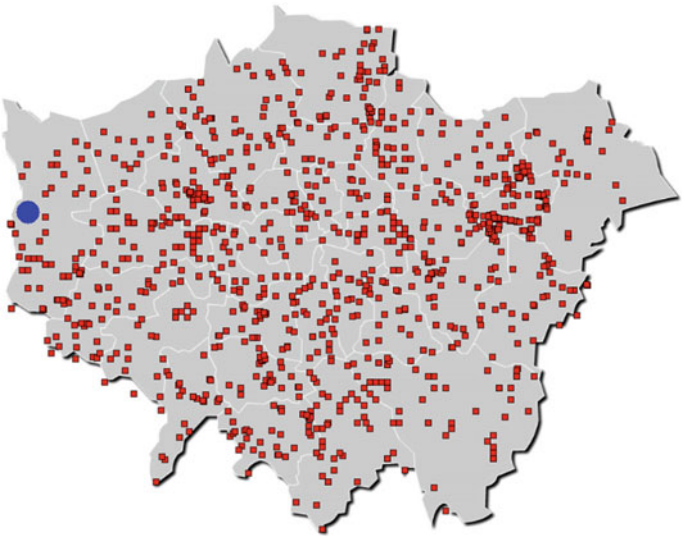


Fig. 9 Pollution sources (squares) and the considered weather station (disk)

6.2.1 Simulation Dataset

We evaluate the deployment model on a data set provided by the Greater London community [40]. London is one of the most polluted cities in Europe [41]. The data set corresponds to the locations of urban pollution sources. In this data set, mostly urban facilities have the potential to affect the air quality such as petrol stations, waste oil burners, cement works. Figure 9 depicts the set of pollution sources, spread over the 32 boroughs of Greater London. Overall, 1090 pollution sources are considered. Pollution sources distribution per borough depends on the surface of the borough and ranges from 6 sources to 161 sources.

In addition to pollution sources locations, we compute the weather scenarios leveraging statistical data gathered by a weather station of London. The location of this station is depicted in Fig. 9. We consider weather conditions of each month of the year averaged over the last past 10 years [42]. The set of weather conditions is depicted in Table 8; each scenario corresponds to values of ambient temperature, wind direction, and wind velocity. As proof of concept and without loss of generality, we assume that weather scenarios provided by the considered weather station are homogeneous in all the area of Greater London.

ILP formulations are implemented using the IBM ILOG CPLEX Optimization Studio and executed on a PC with Intel Xeon E5649 processor under Linux. The ILP solver is executed with a time limit of 30 min. The default values of simulation parameters are summarized in Table 9. We generate the pollution inputs of our deployment model using the Gaussian model presented in Sect. 3 while considering the pollutant characteristics depicted in Table 10. We define the nodes neighboring

Table 8 Weather statistics of London

Month of year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Ambient temperature (°C)	07	07	09	13	15	19	21	20	18	14	09	07
Wind velocity (m/s)	05	05	05	05	05	05	05	05	05	05	05	05
Wind direction (degree)	225	247	270	270	270	225	225	225	225	202	225	247

Table 9 Summary of default simulation parameters

Parameter	Value
Nodes transmission range	100 m
Nodes height	10 m
Sensors cost (c_p^{sensor})	1 monetary unit
Sinks cost (c_p^{sink})	10 monetary units
Coverage requirements of pollution zones (β)	0.98
Coverage requirements of weather conditions (δ)	1.0
Detection sensitivity of sensors ($\mathscr{W}_{ip}^s$)	0.9
Ambient temperature (T)	7 (°C)
Wind velocity (V_w)	5 m/s
Wind direction (D_w)	225°
Pollution threshold (C_0)	20 μg/m^3

Table 10 Common values used for simulation of the Gaussian model

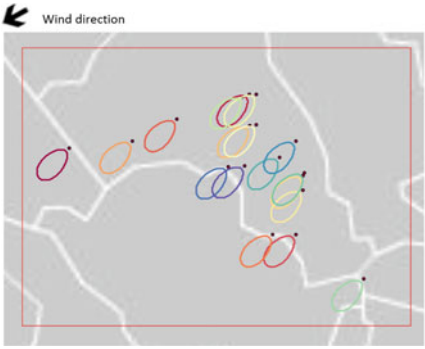
Parameter	Value
h_s	25 m
Q	5 g/s
V	1.9 mm^3/s
T_s	30 °C
a_y	1.36
b_y	0.82
a_z	0.275
b_z	0.69

Γ based on a given transmission range. Moreover, we assume that the cost of nodes is independent of the position of the node, i.e. $c_p^{sensor} = c^{sensor}$ and $c_p^{sink} = c^{sink}$. Furthermore, we investigate the coverage of pollution sources with respect to all the considered scenarios, i.e., $\delta = 1.0$. In addition, we consider that the probabilistic sensing value $\mathscr{W}_{ip}^s$ is constant and equal to 90%.

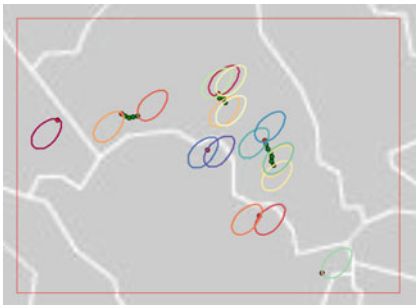
6.2.2 Proof of Concept

As a proof of concept, we first execute the deployment model on the London Borough of Camden. We use streetlights as potential positions of sensors in order to alleviate the energy constraints. The streetlights data set was provided by the Camden DataStore [43]. Camden is spread over an area of around 8 km × 6 km and contains 19 pollution sources. Figure 10 depicts the pollution zones obtained by running of the Gaussian dispersion model while taking into account weather conditions of the

(a) Location of Camden in Greater London



(b) Pollution sources and zones



(c) WSN obtained by application of deployment model. Sensors are depicted in green and sinks in red.

Fig. 10 Application of our deployment model to the London Borough of Camden

month of January. Figure 10 also shows the obtained positions of wireless sensor network nodes computed by the deployment model. We notice that sensors are placed at the intersections of the different pollution zones in order to minimize the coverage deployment cost. Moreover, the resulting network consists of seven sub-networks, a sink is deployed in each one, and some sensors are added to ensure connectivity.

The following results have been obtained by running the deployment model on a hundred of 1200 m $\times$ 1200 m blocks extracted from the Greater London map. The density of pollution sources varies between 3 and 18 sources per block. We discretize each block with a resolution of 100 m to get a $2D$ grid of points that we consider as potential positions of WSN nodes.

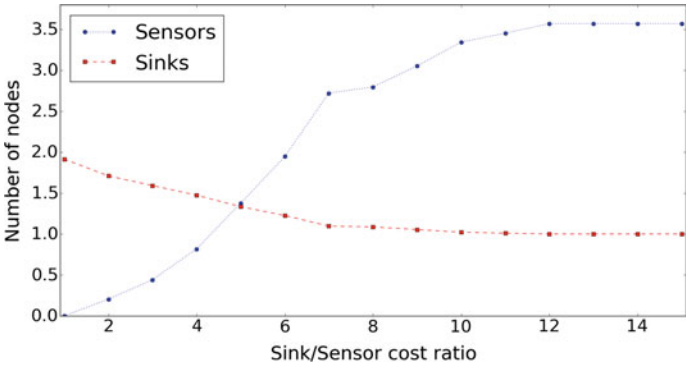


Fig. 11 Average number of sinks and sensors depending on their cost ratio

6.2.3 Evaluation of the Network Connectivity

Evaluation of the number of nodes In this simulation case, we analyze the number of sinks and sensors in the resulting networks while varying the ratio between sink cost and sensor cost. We plot in Fig. 11 the impact of the cost ratio on the optimal number of sensors and sinks. The cost ratio ranges from 1 to 12, and the results are averaged over all the London blocks defined in the previous section. On one hand, we notice that sensors are less used when their cost is close to the sinks cost. For instance, only sinks are used when the cost ratio is equal to 1. On the other hand, when the cost ratio increases, more sensors are used and the number of required sinks tends to one. As a result, the network is usually formed by only one sink when the cost ratio is greater than 10. This is explained by the fact that adding some relay sensor nodes to ensure connectivity has a less cost than using a lot of sinks that are equipped with pollution sensors.

In the following simulations, we execute the deployment model with a default value of sink/sensor cost ratio equal to 10 as shown in Table 9. Thus, we use formulations corresponding to the mono-sink case.

Evaluation of the number of hops to sink nodes In this simulation case, we evaluate the obtained networks in terms of the number of hops to sink nodes, which is a measure of the network lifetime and communication delay [44]. As formulated in our connectivity constraints, the positioning of sink nodes does not take into account the length of sensor-to-sink paths. However, sinks can be relocated on the obtained network when the network is mono-sink, which is the case as shown in the previous subsection. The sink node can be relocated in such a way that the maximum distance to the sink in terms of hops is minimized. This distance is called the radius of the network and describes how much the network is well connected when considering the best position of the sink node. If the sink position is given randomly, the maximum

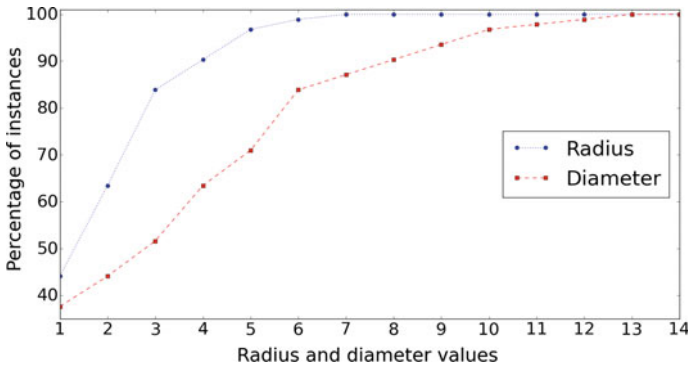


Fig. 12 Cumulative distribution functions of the network radius and diameter

distance to the sink is bounded by the diameter of the network, which is the distance to the sink node when choosing the worst position of the sink.

We depict in Fig. 12 the cumulative distribution functions of the network radius and diameter. Results show that the network radius is as at most equal to 5 for more than 96% of the instances. This means that the number of hops to the sink node after relocation is at most equal to 5 in almost all instances. Moreover, we notice that the network radius is nearly equal to the diameter of the network for the rest of the instances, which means that the sink relocation does not improve much the network connectivity in this case.

6.2.4 Evaluation of the Coverage Results

Impact of nodes height We now study the impact of the height at which are placed sensors and sinks on the deployment cost. We assume that the height of pollution sources is equal to 25 m, and all the sensors and sinks are deployed at the same height, which is considered in the range from 5 to 40 m. We plot in Fig. 13 the sensors and sinks overall deployment cost depending on their height when applying two different weather scenarios, those corresponding to January and December. The results are averaged over all the London blocks. On one hand, we notice that the deployment cost is minimal when the nodes height is close to the effective release height of pollution sources H, which is nearly equal to 25.1 in our case. This is explained by the fact that pollution concentration gets the highest values when being near to the pollutant effective release height H. On the other hand, pollutants are more likely to drop than to increase, which is due to gravitation. Indeed, the deployment cost at 40 m is much greater than the deployment cost at 5 m. Figure 13 also shows that when using different weather scenarios, the deployment cost is not the same. Indeed, weather conditions impact the disposition of pollution zones allowing for more or less intersections. As a result, the obtained WSN topology depends on the weather conditions taken into account.

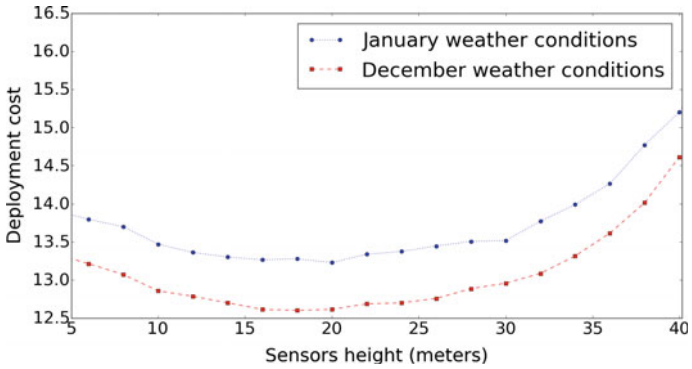


Fig. 13 Deployment cost average depending on nodes height with different weather conditions

Impact of pollution sources density In this scenario, we study the impact of pollution sources density on the deployment cost. For this purpose, we take the results of the previous scenario corresponding to January weather conditions and averaged with respect to the number of pollution sources of each instance, i.e., the number of pollution sources within each block instance. We plot in Fig. 14 the deployment cost variations depending on the nodes height while considering three different densities: four, five, and six pollution sources per instance. Figure 14 shows that more there are pollution sources in the environment, more there are sensors required and thus higher is the deployment cost. This can be explained by the number of pollution zones that increases with the number of pollution sources and thus requires much sensors to ensure the coverage requirements. In addition, the increasing in the deployment cost from five sources density to six sources density is less than the increasing from four sources density to five sources density. This is because when the number of pollution sources increases, more intersections between pollution zones appear and affect the increasing of the deployment cost.

Impact of probabilistic sensing The probabilistic sensing of pollution sensors is one of the most important factors that affect the topology of sensor networks used for pollution monitoring. Figure 15 depicts the average cost of the resulting deployments of the block instances while considering two values of the detection sensitivity of sensors: $\mathscr{W}_{ip}^{s} = 0.9$ and $\mathscr{W}_{ip}^{s} = 0.8$. As expected, using sensors with better detection sensitivity yields less deployment cost. We notice that the ratio between the two curves is around 1.1. This is explained by the intersection existence between the different polluted zones, which means that in some cases a sensor can monitor more than one pollution source.

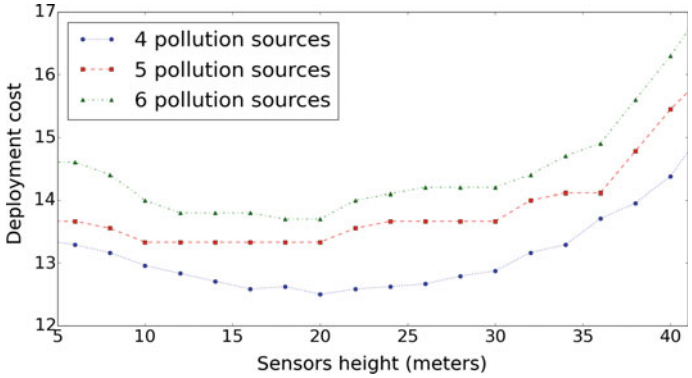


Fig. 14 Deployment cost average depending on nodes height and pollution sources density

Impact of the number of weather conditions In this simulation case, we study the impact of using a small number of weather scenarios on the deployment results. It is clear that when considering all the possible weather scenarios, the resulting WSN ensures better pollution monitoring. However, when there is a huge number of weather scenarios, considering a less number of these scenarios alleviates the deployment model allowing their application on large-scale instances.

We recall that $\mathscr{S}$ is the set of the monthly weather scenarios presented in Table 8. Given a subset $\mathscr{S}'$ of $\mathscr{S}$, we define the missed pollution zones percentage as the percentage of pollution zones that cannot be covered by the WSN resulting from executing the model under only weather scenarios S'. Figure 16 illustrates the variations of the missed pollution zones percentage depending on the number of weather scenarios taken into account starting from January weather scenario in the first curve

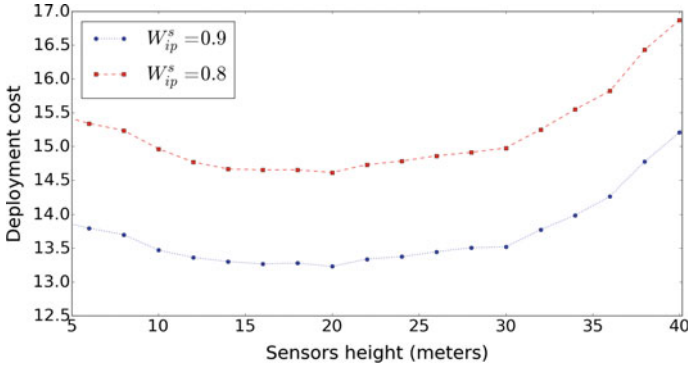


Fig. 15 Deployment cost average depending on nodes height and probabilistic sensing values

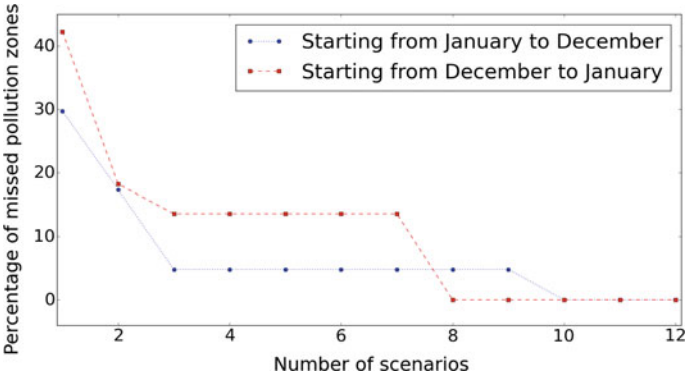


Fig. 16 Average percentage of missed pollution zones depending on weather scenarios

and starting from December weather scenario is the second one. Figure 16 shows that the percentage of missed pollution zones usually decreases when considering more weather scenarios; this is expected since the number of pollution zones depends on the weather scenarios set. However, in some cases, the missed pollution zones percentage remains the same when considering additional scenarios of weather conditions. Indeed, additional weather scenarios involve new pollution zones that may be, in some cases, already included in the set of pollution zones formed without taking into account the additional scenarios. This may happen, for instance, when additional weather scenarios are slightly different from those already taken into account.

In addition to the impact of the number of weather scenarios, their similarity has also to be taken into account. As shown in Fig. 16, considering only weather scenarios from December to May meaning only eight scenarios allows to cover the whole set of pollution zones in contrary to the scenarios set from January to October that requires ten scenarios.

7 Discussion and Future Directions

In this chapter, we focused on what the literature lacks in WSN deployment for air pollution monitoring, which is an appropriate coverage formulation of both regular mapping and high concentrations detection. We have also shown how connectivity and coverage can be merged into one constraint in order to make the deployment models less complex. More details about the three presented models can be found, respectively, in [45–48].

Since the proposed models are integer linear programs, their complexity mainly depends on the number of binary variables which represent the potential positions of sensor nodes. Therefore, the execution time of solvers increases exponentially with the area of the deployment region (if we keep the same spatial resolution of maps). As

for the number of pollution snapshots which characterize weather scenarios, it mainly impacts the number of constraints. Hence, the execution time increases linearly with the number of time frames.

We believe that our models can be extended in order to take into account the network lifetime and mobile deployment. We provide hereafter some details about how the extension of our models can be achieved.

Optimization of the network lifetime One possible solution is to base on the work of [12] where authors define the energy constraints based on the flow concept. This solution is viable since our models are also based on the flow concept. The idea is to split the lifetime of the network into a sequence of time frames (a sequence of minutes for instance) and enforce each sensor node to send a flow unit in each time frame. Then, a new constraint should be added to the models in order to ensure that for each sensor, the sum of the energy that is consumed in the set of time frames is less than the energy of the battery of the sensor. The lifetime objective function corresponds to the maximum number of time frames where the network is operating.

Mobile deployment Indeed, our models are designed for static networks, which is argued by the fact that pollution sensors operate well when they are static [49]. However, sinks can be considered as mobile nodes, and this can be integrated into our models based on existing mobile-sinks formulations such as the work of [12]. The idea is to consider a set of time frames as in the extension of network lifetime. The mobile sink changes its positions in each time frame. Then, the flow constraints should be formulated in order to ensure that the flow is conservative in the network in each time frame. This means that in each time frame, each sensor generates a flow unit and the sink node receives all the generated units.

8 Conclusion

Air pollution is becoming a major problem of smart cities due to the increasing industrialization and the massive urbanization. In this chapter, we focused on the use of wireless sensor networks for the two main applications of air pollution monitoring: regular mapping and the detection of threshold crossings. We addressed the deployment issue and presented three optimization models ensuring pollution coverage and network connectivity with the minimum cost. Unlike the inadequate related works, which are either generic or rely on a detection model, we based on the nature of the phenomenon in order to provide realistic models.

We evaluated the impact of the parameters of the different models on the deployment results while using real data sets. Among our conclusions are the fact that sensors should be placed at a height close to the one of pollution sources and the fact that the size of the resulting network depends on the degree of the variations of pollution concentrations within the deployment region. We also concluded that the desired monitoring precision impacts well the density of sensors and hence the connectivity of the network.

As future work, we are trying to consider the impact of the nature of pollutants and the urban topography on the coverage results. We are also working on the validation of our deployment models.

Acknowledgements This work has been supported by the "LABEX IMU" (ANR-10-LABX-0088) and the "Programme Avenir Lyon Saint-Etienne" of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR).

References

1. World Health Organization: Burden of Disease from Household Air Pollution for 2012. http://www.who.int/phe/health_topics/outdoorair/databases/FINAL_HAP_AAP_BoD_24March2014.pdf. Accessed 27 Feb 2016
2. AirParif: The Air Quality Monitoring Organization of the Paris Agglomeration. http://www.airparif.fr/en/index/index
3. Kumar, A., Kim, H., Hancke, G.P.: Environmental monitoring systems: a review. Sens. J. IEEE **13**(4), 1329–1339 (2013)
4. Mead, M., Popoola, O., Stewart, G., Landshoff, P., Calleja, M., Hayes, M., Baldovi, J., McLeod, M., Hodgson, T., Dicks, J., et al.: The use of electrochemical sensors for monitoring urban air quality in low-cost, high-density networks. Atmos. Environ. **70**, 186–203 (2013)
5. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Comput. Netw. **38**(4), 393–422 (2002)
6. ETSI: Electromagnetic Compatibility and Radio Spectrum Matters (ERM); System Reference Document (SRdoc): Spectrum Requirements for Short Range Device, Metropolitan Mesh Machine Networks (M3N) and Smart Metering (SM) Applications. http://www.etsi.org/deliver/etsi_tr/103000_103099/103055/01.01.01_60/tr_103055v010101p.pdf. Accessed 20 Jan 2016
7. Liu, B., Dousse, O., Nain, P., Towsley, D.: Dynamic coverage of mobile sensor networks. IEEE Trans. Parallel Distrib. Syst. **24**(2), 301–311 (2013)
8. Zhu, C., Zheng, C., Shu, L., Han, G.: A survey on coverage and connectivity issues in wireless sensor networks. J. Netw. Comput. Appl. **35**(2), 619–632 (2012)
9. Chakrabarty, K., Iyengar, S.S., Qi, H., Cho, E.: Grid coverage for surveillance and target location in distributed sensor networks. IEEE Trans. Comput. **51**(12), 1448–1453 (2002)
10. Altınel, İ.K., Aras, N., Güney, E., Ersoy, C.: Binary integer programming formulation and heuristics for differentiated coverage in heterogeneous sensor networks. Comput. Netw. **52**(12), 2419–2431 (2008)
11. Chang, X., Tan, R., Xing, G., Yuan, Z., Lu, C., Chen, Y., Yang, Y.: Sensor placement algorithms for fusion-based surveillance networks. IEEE Trans. Parallel Distrib. Syst. **22**(8), 1407–1414 (2011)
12. Keskin, M.E., Altınel, İ.K., Aras, N., Ersoy, C.: Wireless sensor network lifetime maximization by optimal sensor deployment, activity scheduling, data routing and sink mobility. Ad Hoc Netw. **17**, 18–36 (2014)
13. Rebai, M., Snoussi, H., Hnaien, F., Khoukhi, L., et al.: Sensor deployment optimization methods to achieve both coverage and connectivity in wireless sensor networks. Comput. Oper. Res. **59**, 11–21 (2015)
14. Sengupta, S., Das, S., Nasir, M., Panigrahi, B.K.: Multi-objective node deployment in wsns: In search of an optimal trade-off among coverage, lifetime, energy consumption, and connectivity. Eng. Appl. Artif. Intell. **26**(1), 405–416 (2013)

15. Guney, E., Altinel, I., Aras, N., Ersoy, C.: Efficient integer programming formulations for optimum sink location and routing in wireless sensor networks. In: 23rd International Symposium on ISCIS'08. IEEE, pp. 1–6 (2008)
16. Güney, E., Aras, N., Altınel, İ.K., Ersoy, C.: Efficient integer programming formulations for optimum sink location and routing in heterogeneous wireless sensor networks. Comput. Netw. **54**(11), 1805–1822 (2010)
17. Roy, V., Simonetto, A., Leus, G.: Spatio-temporal sensor management for environmental field estimation. Signal Proc. **128**, 369–381 (2016)
18. Krause, A., Singh, A., Guestrin, C.: Near-optimal sensor placements in Gaussian processes: theory, efficient algorithms and empirical studies. J. Mach. Learn. Res. **9**(Feb), 235–284 (2008)
19. Krause, A., Guestrin, C., Gupta, A., Kleinberg, J.: Robust sensor placements at informative and communication-efficient locations. ACM Trans. Sens. Netw. (TOSN) **7**(4), 31 (2011)
20. Wu, X., Liu, M., Wu, Y.: In-situ soil moisture sensing: Optimal sensor placement and field estimation. ACM Trans. Sens. Netw. (TOSN) **8**(4), 33 (2012)
21. Dang, T., Frolov, S., Bulusu, N., Feng, W.C., Baptista, A.: Near optimal sensor selection in the Columbia river (Corie) observation network for data assimilation using genetic algorithms. In: International Conference on Distributed Computing in Sensor Systems (DCOSS). Springer, pp. 253–266 (2007)
22. Liaskovitis, P.G., Schurgers, C.: Leveraging redundancy in sampling-interpolation applications for sensor networks: a spectral approach. ACM Trans. Sens. Netw. (TOSN) **7**(2), 12 (2010)
23. Ranieri, J., Chebira, A., Vetterli, M.: Near-optimal sensor placement for linear inverse problems. IEEE Trans. Signal Proc. **62**(5), 1135–1146 (2014)
24. Du, W., Xing, Z., Li, M., He, B., Chua, L.H.C., Miao, H.: Sensor placement and measurement of wind for water quality studies in urban reservoirs. ACM Trans. Sens. Netw. (TOSN) **11**(3), 41 (2015)
25. Khalfallah, Z., Fajjariy, I., Aitsaadiz, N., Langar, R., Pujolle, G.: A new WSN deployment algorithm for water pollution monitoring in Amazon rainforest rivers. In: Global Communications Conference (GLOBECOM). IEEE, pp. 267–273, (2013)
26. Marshall, J.D., Nethery, E., Brauer, M.: Within-urban variability in ambient air pollution: comparison of estimation methods. Atmos. Environ. **42**(6), 1359–1369 (2008)
27. Jerrett, M., Arain, A., Kanaroglou, P., Beckerman, B., Potoglou, D., Sahsuvaroglu, T., Morrison, J., Giovis, C.: A review and evaluation of intraurban air pollution exposure models. J. Expo. Sci. Environ. Epidemiol. **15**(2), 185–204 (2005)
28. Daly, A., Zannetti, P.: Air pollution modeling–an overview. Ambient Air Pollut. (2007)
29. Stockie, J.M.: The mathematics of atmospheric dispersion modeling. Siam Rev. **53**(2), 349–372 (2011)
30. Wong, D.W., Yuan, L., Perlin, S.A.: Comparison of spatial interpolation methods for the estimation of air quality data. J. Expo. Sci. Environ. Epidemiol. **14**(5), 404–415 (2004)
31. Hoek, G., Beelen, R., De Hoogh, K., Vienneau, D., Gulliver, J., Fischer, P., Briggs, D.: A review of land-use regression models to assess spatial variation of outdoor air pollution. Atmos. Environ. **42**(33), 7561–7578 (2008)
32. Gallart, V., Felici-Castell, S., Delamo, M., Foster, A., Perez, J.J.: Evaluation of a real, low cost, urban WSN deployment for accurate environmental monitoring. In: 2011 IEEE 8th International Conference on Mobile Ad Hoc and Sensor Systems (MASS). IEEE, pp. 634–639 (2011)
33. Soulhac, L., Salizzoni, P., Mejean, P., Didier, D., Rios, I.: The model sirane for atmospheric urban pollutant dispersion; part ii, validation of the model on a real case study. Atmos. Environ. **49**, 320–337 (2012)
34. Tilloy, A., Mallet, V., Poulet, D., Pesin, C., Brocheton, F.: Blue-based no2 data assimilation at urban scale. J. Geophys. Res. Atmos. **118**(4), 2031–2040 (2013)
35. Air Rhône-Alpes: The Air Quality Monitoring Organization Of The Lyon Agglomeration. http://www.air-rhonealpes.fr. Accessed 27 Jan 2016
36. Shekhar, S., Kang, J., Gandhi, V.: Spatial data mining. In: Liu, L., Özsu, M. (eds.) Encyclopedia of Database Systems. Springer US, pp. 2695–2698 (2009)

37. Kama, A.A.L., Hammadou, H., Meurisse, B., Papaix, C.: Le pic de pollution a paris du 12 au 17 mars 2014. Tech. Rep
38. Cardei, M., Pervaiz, M.O., Cardei, I.: Energy-efficient range assignment in heterogeneous wireless sensor networks. In: International Conference on ICWMC'06. IEEE, pp. 11–11 (2006)
39. Patel, M., Chandrasekaran, R., Venkatesan, S.: Energy efficient sensor, relay and base station placements for coverage, connectivity and routing. In: 24th IEEE International on IPCCC. IEEE, 581–586 (2005)
40. London DataStore: London atmospheric emissions inventory 2010. http://data.london.gov.uk/dataset/london-atmospheric-emissions-inventory-2010. Accessed 15 Jan 2016
41. The guardian: London Ranks Among Worst European Cities for Air Pollution. http://www.theguardian.com/environment/2011/sep/07/london-worst-european-cities-air-pollution. Accessed 15 Feb 2016
42. Wind Finder: Wind and Weather Statistics London-Heathrow. http://www.windfinder.com/windstatistics/london-heathrow. Accessed 15 Jan 2016
43. Camden Data: Camden Lighting Point. http://www.camdendata.info/Pages/Home.aspx. Accessed 15 July 2015
44. Gandham, S.R., Dawande, M., Prakash, R., Venkatesan, S.: Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In: Global Telecommunications Conference on GLOBECOM'03, vol. 1, pp. 377–381. IEEE(2003)
45. Boubrima, A., Bechkit, W., Rivano, H.: Error-bounded air quality mapping using wireless sensor networks. In: 2016 IEEE 41st Conference on Local Computer Networks (LCN). IEEE, pp. 380–388 (2016)
46. Boubrima, A., Bechkit, W., Rivano, H.: A new WSN deployment approach for air pollution monitoring. In: 2017 IEEE 14th Conference on Consumer Communications and Networking (CCNC). IEEE
47. Boubrima, A., Matigot, E., Bechkit, W., Rivano, H., Ruas, A.: Optimal deployment of wireless sensor networks for air pollution monitoring. In: 24th International Conference on Computer Communications and Networks (ICCCN'15). IEEE, pp. 1–7 (2015)
48. Boubrima, A., Bechkit, W., Rivano, H.: Optimal WSN deployment models for air pollution monitoring. IEEE Trans. Wirel. Commun (To appear)
49. Velasco, A., Ferrero, R., Gandino, F., Montrucchio, B., Rebaudengo, M.: A mobile and low-cost system for environmental monitoring: a case study. Sensors **16**(5), 710 (2016)

Mobile Coverage

Hyunbum Kim

Abstract In wireless sensor networks, a *coverage* is largely classified into three categories: *area coverage*, *barrier coverage*, *sweep coverage*. The *area coverage* is to perform a static deployment of sensors with a possible use of mobility of sensors so as to maximize the total area. The *barrier coverage* is to construct barriers which are able to detect penetrations of intruders. The *sweep coverage* is to monitor specific points of interests (PoI) periodically where the coverage at each PoI should be time-variant. This chapter deals with those coverage categories based on the mobility of sensors. Then, for each coverage, we focus on introducing various research problems and critical issues by mobile sensors including how mobile sensors can be applied to the problems in the coverage area in order to achieve specific objectives defined by systems. Also, we describe various strategies and their solutions by those novel approaches in the specific area.

1 Area Coverage Using Mobility

The *area coverage* is to focus on maximizing the total given area in wireless sensor networks [1–5]. A mobility of sensors can be utilized in the *area coverage*. We will review several issues and problems under the *area coverage*.

1.1 Area Coverage Problem by Mobile Sensor Deployment Using Potential Fields

Howard et al. [7] introduced the deployment problem of mobile sensors within an unknown environment. Each mobile sensor has abilities of sensing, computation, communication, and locomotion. In particular, the locomotion allows various useful network features including *self-deploy*, which is starting from initial configuration

H. Kim (✉)
University of North Carolina at Wilmington, Wilmington, NC, USA
e-mail: kimh@uncw.edu

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_5

and the sensors in the network can spread out so that the covered area is maximized. Howard et al. [7] proposed a *potential-field-based* approach in which sensors are considered as virtual particles.

1.1.1 Potential Fields

Without global maps, Howard et al. [7] considered the concept of *potential fields* which were used in mobile robotics. The fields can be constructed by the way that each sensor is repelled by obstacles and by other sensors in order to allow sensors to spread out throughout the environment.

Then, the basic potential field scheme is as follows.

A sensor is subject to a force F which is a gradient of scalar potential field U. So, we have:

$$F = - \bigtriangledown U \tag{1}$$

Then, we consider the potential field into two parts: one field U_o for obstacles and another field U_n for other nodes. Note that these fields allow repulsive forces F_o and F_n. It follows that $U = U_o + U_n$ and $F = F_o + F_n$. Assume that k_o is a constant for the strength of the field U_o and r_i is the Euclidean distance between the sensor and obstacle i. If we consider that each sensor and each obstacle takes an electric change, then the resultant *electrostatic* potential can be:

$$U_o = k_o \sum_i \frac{1}{r_i} \tag{2}$$

The result is for all obstacles which can be seen by the sensor k_o.

Suppose that x is a location of the sensor and x_i is a location of obstacle i. r_i can be computed as $r_i = |x_i - x|$ and $r_i' = x_i - x$. Then, the total force F_o can be:

$$F_o = -\frac{dU_o}{dx} = -\sum_i \frac{dU_o}{dr_i} \cdot \frac{dr_i}{dx} \tag{3}$$

With this, we can obtain the following derivation:

$$F_o = -k_o \sum_i \frac{1}{{r_i}^2} \cdot \frac{r_i'}{r_i} \tag{4}$$

Also, we note the force is for the relative positions r_i of obstacles rather than their absolute locations x_i.

On the other hand, let us consider the potential field U_n for other nodes. Then, with r_i is the relative location of sensor i, we can express the potential U_n and force F_n as follows:

$$U_n = -k_n \sum_i \frac{1}{r_i} \tag{5}$$

$$F_n = -k_n \sum_i \frac{1}{{r_i}^2} \cdot \frac{r_i'}{r_i} \tag{6}$$

1.1.2 The Equation of Motion and the Control Law

Assume that $\ddot{x}$ is an acceleration of the sensor, m is its mass, v is the viscosity coefficient, and $\dot{x}$ is the sensor velocity. We describe that a trajectory of the sensor (subject to force F) is calculated based on *Equation of Motion* as follows:

$$\ddot{x} = \frac{(F - v\dot{x})}{m} \tag{7}$$

A *virtual physical system* can be mapped into a *real physical system* consisting of real nodes. These sensors will construct a form of velocity controller. It follows that the mapping from virtual to real physical system can be performed by a *control law* mapping a virtual force onto a velocity *control vector*. Assume that v' is the commanded velocity at some time t and $\triangle v'$ is the change in the commanded velocity between t and $t + \triangle t$. Then, the change in commanded velocity is computed as follows:

$$\triangle v' \leftarrow \frac{(F - vv')}{m \cdot \triangle t} \tag{8}$$

With the assumption that a_{max} is the biggest allowable change of velocity, we can consider that x and y components of $\triangle v'$ are clipped on condition that $-a_{max} \leq \triangle v' \leq a_{max}$. Then, the commanded velocity v' is decided as follows:

$$v' \leftarrow v' + \triangle v' \tag{9}$$

Similarly, it is clipped on condition that $-v_{max} \leq v' \leq v_{max}$ where v_{max} is the largest allowable velocity.

1.1.3 Area Coverage with a Time Interval

In [8], Liu et al. studied the area coverage of both stationary and mobile sensor networks. They defined area coverage, area coverage over a time interval formally as follows.

Definition 1 (*Area coverage*) At time t, the area coverage $f_a(t)$ is the fraction of the geographical region which is covered by at least one sensor node.

Definition 2 (*Area coverage over a time interval*) During time interval $[0, t)$, $f_i(t)$ is the fraction of the geographical region which are covered by at least one sensor node within $[0, t)$.

In [8], Liu et al. considered the density of the Poisson point process as λ. Because a sensor covers a range r, the initial configuration of the network is expressed by a Poisson Boolean model $B(\lambda, r)$. Based on the definition and settings, Liu et al. [8] proposed the following Theorem 1 about the stationary sensor network.

Theorem 1 *At any given time instant $t > 0$, the area coverage for a stationary network with $B(\lambda, r)$ is $f_a(t) = 1 - e^{-\lambda\pi r^2}$*

For the Theorem 1, the proof is shown by [8, 9] as follows.

Proof Assume we have a bounded area R and the vacancy V within R is the region in R not covered by sensor nodes.

$$V = \int_R (\chi(x)dx, \text{where } \chi(x) \text{ is 1if } x \text{ is not covered. Otherwise, } \chi(x) \text{ is 0.}$$

By Fubini's theorem, we get $E(V) = \int_R (\chi(x))dx$. We assume there is an arbitrary point x within R and the number of sensors that covers x is N. Also, let us assume that the point x is covered by at least one sensor within r. It follows that N is able to have Poisson distribution with $\lambda\pi r^2$. So, we have the following equations.

$$E\{(\chi(x))\} = P(x\text{is not covered}) = P(N = 0) = e^{-\lambda\pi r^2}.$$

And, we have $E(V) = \int_R E\{(\chi(x))\}dx = \|R\|e^{-\lambda\pi r^2}$.
Since the derivation is independent from R, we have the following area coverage.

$$f_a = 1 - \frac{E(V)}{\|R\|} = 1 - e^{-\lambda\pi r^2}.$$

For stationary sensor networks, a point within a region has two cases: it is covered or not covered. Note that the area coverage does not change during time.

Liu et al. [8] also considered a sensor mobility. Figure 1 depicts the effect of sensor mobility about area coverage. Figure 1a represents the initial network configuration at time 0. The solid disks show the area being covered at the given time instant. Also, Fig. 1b shows the effect of sensor mobility during time interval $[0, t)$. The union of the shared area and the solid disks depicts the region being covered during the time interval $[0, t)$.

Then, for effect by sensor mobility on the area coverage, Liu et al. [8] proposed the following Theorem 2.

Theorem 2 *Assume that we have a sensor network $B(\lambda, r)$ at time $t = 0$ such that sensors are moving by the random mobility model.*

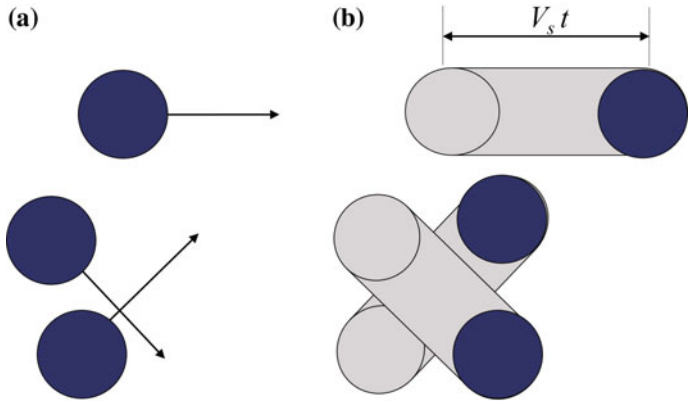


Fig. 1 Coverage of mobile sensor network with a time interval

1. *At any given time instant $t > 0$, the fraction of the covered area is*

$$f_a(t) = 1 - e^{-\lambda \pi r^2}, \text{ where } \forall t \geq 0.$$

2. *The fraction of area which has been covered at least once during interval* $[0, t)$ *is*

$$f_i(t) = 1 - e^{-\lambda(\pi r^2 + 2rE|V_s|t)}$$

To prove Theorem 2, Liu et al. [8] showed the below proof.

Proof Assume that sensor nodes have initial locations and we apply the random mobility model. At any time instant t, the positions of the sensor nodes generate two-dimensional Poisson point process. Then, the ration of the region covered at time t is still equivalent to in the initial setting, $f_a(t) = 1 - e^{-\lambda \pi r^2}$. By Fig. 1, each sensor nodes covers a shape of a racetrack whose expected region is

$$\alpha = E[\pi r^2 + 2rV_s t] = \pi r^2 + 2rE[V_s]t,$$

where $E[v_s] = \int_0^{V_{max}} f_V^s(V) dV$ is considered as the expected sensor speed.

Moreover, the area coverage depends on the distribution of the random shaped only by its expected region. Hence, we can derive

$$f_i(t) = 1 - e^{-\alpha\lambda} = 1 - e^{-\lambda(\pi r^2 + 2rE|V_s|t)}.$$

1.2 Constrained Coverage by Mobile Sensors

The mobile sensors in the network can be used for various objectives such as collection of data, repairing the static sensors. Also, for those objectives, the deployment strategy is important because it can provide sensor coverage to meet specific local constraint (i.e., node degree) and global constraint (i.e., network connectivity). Also, a *constrained coverage* is the problem to find a deployment formation that maximizes the collective coverage by sensors such that constraints are satisfied. Based on the observation, Poduri et al. [10] considered the *constrained coverage* by autonomous mobile sensors such that the constraint of node degree is satisfied.

1.2.1 *Constrained Coverage* Problem

Assume that we have no global positioning system (GPS). Also, let us assume sensors are able to to do omnidirectional motion and each sensor can verify the exact relative range and its neighbors. Suppose that the quality of sensing (communication) is constant within R_s (R_c) and is zero outside the sensing range. We also define two sensors have a neighbor relationship if the Euclidean distance between them is at most the communication range R_c. With this setting, we define the *constrained coverage* problem as follows.

Definition 3 (*constrained coverage*) Given a set of mobile sensors N with isotropic radial sensor range R_s and isotropic radio communication range R_c, the *constrained coverage* problem is to deploy sensors so that the deployed formation is to maximize the sensor coverage of the network satisfying the constraint that each sensor has at least k neighbors.

Also, the *normalized per-node coverage* can be defined as:

$$coverage = \frac{\text{(covered area by network)}}{N\pi {R_s}^2} \tag{10}$$

Theorem 3 *The OSPP is NP-complete [11].*

For Theorem 3, we prove *OSPP* is NP-complete.

1.3 Cooperative Dynamic Coverage by Static Sensors and Mobile Sensors

The objective of mobile sensors is to maximize some metric of information such as monitoring coverage, probability of event detection while we have constraints on energy or detection time. Lambrou et al. [11] considered the use of mobile sensors

to enhance the area coverage for the environment of sparse static sensors. With the mixed use of static sensors and mobile sensors, Lambrou et al. [11] introduced an *OSPP* (*Optimal Search Path* problem) that is to find the optimal path to maximize the dynamic area coverage.

1.3.1 Dynamic Area Coverage

Suppose that we consider a sparse sensor network with a numerous number of static sensors and a few mobile sensors, which are deployed in the square-shaped area. Let us assume that we have a set of static sensors S with $S_{num} = |S|$ which are randomly deployed in area A where each sensor is located at $\hat{x_i} = (x_i, y_i)$, $i = 1, 2, ..., S_{num}$. Also, we have a set of mobile sensors M with $M_{num} = |M|$ and its position after the kth time step is $\hat{x_i} = (x_i(k), y_i(k))$, $i = 1, ..., M_{num}$, $k = 0, 1, ...$ Also, assume each sensor has a known sensing range r_d and a known communication range $r_c > r_d$. Every sensor is able to own location using a GPS and a localization. Figure 2 shows an example of mixed sensor network model with static sensors and mobile sensors with a description of r_d and r_c. In particular, we define the set of all sensor nodes $N = S \cup M$ and the total number of sensors is $N_{num} = S_{num} + M_{num}$. And, a sensor s has neighbors which are positioned at a distance less than or equal to r_c from s. With the assumption that $\| \cdot \|$ means the *Euclidean norm*, the neighbors of s can be denoted as follows:

$$H_{r_c}(s) = \{j : \|\hat{x_s} - \hat{x_j}\| \leq r_c, j \in N, j \neq s\} \tag{11}$$

We also define a static point event $\hat{e} = (x^e, y^e)$ in A and the event r emits a signal, which can be detected by near sensors. Its form is as follows:

$$s^e(\hat{x}, t) = I(\|\hat{x} - \hat{e}\| \leq r_d) \cdot (u(t - t_e^{ON}) - u(t - t_e^{OFF})), \tag{12}$$

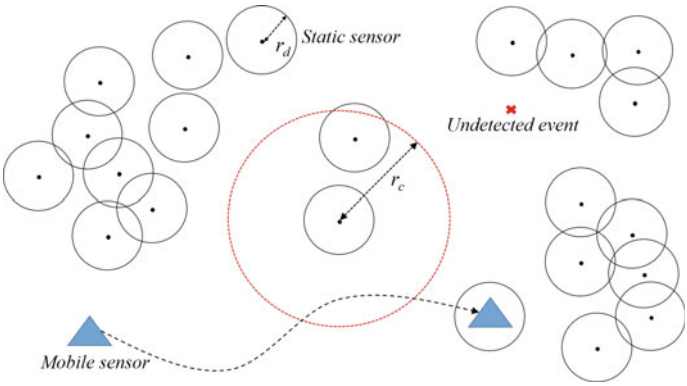


Fig. 2 An example of mixed sensor network model

where $\hat{x} \in A$ and $I(\|\hat{x} - \hat{e}\| \leq r_d)$ is the indicator function that has 1 if the condition $\|\hat{x} - \hat{e}\| \leq r_d$ is satisfied or 0, otherwise; also, $u(t)$ is the step function; and t_e^{ON}, t_e^{OFF} are the times at which the event is turned ON and OFF. If the event occurs within the sensing range of static sensor, it is detected immediately. But, if the event occurs at a point that is not covered by any sensor, then it should be undetected. Therefore, the mission of every mobile node is to sample the uncovered areas on condition that the event within an uncovered area is detected completely with a possible minimal time.

Then, we define the *dynamic area coverage*, referred as C, as follows.

Definition 4 (*dynamic area coverage*) The *dynamic area coverage* is an objective function to be maximized by the mobile sensors. At any instant t, suppose $I(\hat{x}, t)$ is an indicator function which returns 1 if the point $x \in A$ is monitored by at least one static or mobile sensor within the interval $[0, t]$ and returns 0 otherwise. It follows that $I(\hat{x}, t) = 1$ if there is a sensor $s \in S$ on condition that $\|\hat{x_s} - \hat{x}\| \leq r_d$ or if a sensor $s \in M$ passes from the point covered by $\hat{x}$, then $\|\hat{x_s(k)} - \hat{x}\| \leq r_d$, where $k \cdot \delta t \leq t$ and δt is the sampling period.

Then, the coverage by the network at t can be expressed as:

$$C(t) = \frac{1}{A} \int_A I(\hat{x}, t) d\hat{x} \tag{13}$$

With the assumption that the event occurs at any point of the area with equal probability, $C(t)$ defines the probability $P(t)$ that a static event is monitored by at least one sensor within time interval $[0, t]$, where $t \leq T$ and T is the needed time of mobile sensors to provide full coverage of the uncovered areas. The *uncovered area* is the set of points which are not covered by any sensor. It can be defined as:

$$U(t) = \{\hat{x} : I(\hat{x}, t) = 0\} \tag{14}$$

The *uncovered area* may be composed of one ore more connected subsets. Then, each connected subset is called as *coverage hole*.

Because mobile sensor move, they will cover new areas. Hence, a reasonable objective function to be maximized by mobile sensors is also considered as *dynamic area coverage*. It can be formulated as:

$$C(T) = \int_0^T C(t) dt \tag{15}$$

It follows that for a given T, $C(T)$ is to be maximized when the best trajectories by mobile sensors return the best rate of coverage $C(t)$ over time.

1.3.2 *OSPP* Problem

Now, we define *OSPP* (Optimal Search Path Problem) is as follows.

Definition 5 (*OSPP*) The *OSPP* is to find the optimal path which maximizes the *dynamic area coverage* $C(T)$ when T is given.

Theorem 4 *The OSPP is NP-complete [11].*

For Theorem 4, we prove *OSPP* is NP-complete.

Proof Firstly, we show that $OSPP \in NP$. Assume that we use the sequence of h coverage holes found in the path as a certificate. The verification algorithm checks that the sequence includes every hole and sums up the costs including Euclidean distance interhole movement and hole searching and calculates if the sum is to be minimum. This step can be done in polynomial time.

In order to prove that *OSPP* is NP-hard, we do a reduction from an arbitrary instance of Euclidean path TSP (*EpTSP*), which is a well-known NP-hard problem by [12] to a special instance of *OSPP*. Given the *EpTSP* instance (h, d_{ij}) where h is the number of cities and d_{ij} means the matrix for distance between cities, we choose that $M = 1$ and this single mobile sensor is positioned in an arbitrary coverage hole initially. Then, the static sensor deployment is such that it enables a definite number of h holes and the area of each $A_i, i = 1, 2, ..., h$ is set to $A_i = 0$ and we have finally $r_d = 0$. Therefore, the optimal trajectory of mobile sensor is the path that visits every isolated uncovered points with the minimal distance. Because of these choices, the optimal solution of the special case of *OSPP* problem will coincide with the optimal solution of the *EpTSP*. So, we prove that *OSPP* is NP-complete. □

2 Barrier Coverage Using Mobility

A full coverage in wireless sensor networks monitor a whole area continuously. So, it is guaranteed that any point in the given area is monitored by at least one or k sensors. On the other hand, *barrier coverage* is a special type of partial coverage. The concept of *barrier coverage* was initially introduced by Gage [6]. In the work, they considered the barrier coverage based on robotic sensors. Then, in [13], Kumar et al. introduced the notion of k-barriers using sensor nodes. With the construction of k-barriers, the system can guarantee that an intruder's penetration from one side to the other side is detected by at least k distinct sensor nodes. Since it is proper for various important applications such as border protection and intrusion detection [13–15] and it has an advantage of reducing the number of sensors to perform the detection of the penetration when compared with full coverage, the *barrier-coverage* has been one of critical research areas in wireless sensor networks.

2.1 Barrier Coverage to Detect Mobile Objects

Basically, k-barriers were introduced by Kumar et al. [13]. Those k-barriers are able to provide that at least k distinct sensors detect any intruders penetration which moves from one side to the other side in the given square-shaped area. Also, weak and strong barriers were defined by Liu et al. [16] and Li et al. [17]. Figure 3 describes the concept of weak barrier in the given area. Suppose each circle depicts the sensing range of sensor. Then, most orthogonal crossing paths (dotted lines) from top to bottom are detected by at least one sensor. But, there is the possibility that uncovered paths exist (see the solid path in Fig. 3). So, the barrier status is defined as *weak barrier*. On the other hand, Fig. 4 shows the *strong barrier*. Also, suppose each circle depicts the sensing range of sensor. The constructed barrier by shaded circles provides a strong barrier such that no penetration of intruders (i.e., dotted lines) can cross the region undetected.

Also, *local barrier* was introduced by Chen et al. [18]. Based on the observation on which movements are highly likely to pursue a shorter path in crossing a given region, the *local barrier* guarantees the detection of every intruder movement whose penetration trajectory is restricted to a slice of the belt region. The proponents showed that each sensor is able to decide the existence of local barrier that the region of interest is temporarily curved. And, for some intrusion detection applications, only one direction of crossing area could be illegal. *One-way barrier* considers such a situation. So, if a given area is covered by one-way barrier, the *one-way barrier* systems will detect the penetration if and only if an attacker is crossing with pre-specified direction. Figure 5 represents *one-way barrier*. Suppose the sensing range of each sensor is depicted as the circle. Then, the shaded *one-way barrier* can trigger an alarm for the penetration by pre-specified direction. As seen in Fig. 5, crossing trajectory (solid lines) from bottom to top is detected as illegal. But, the movement

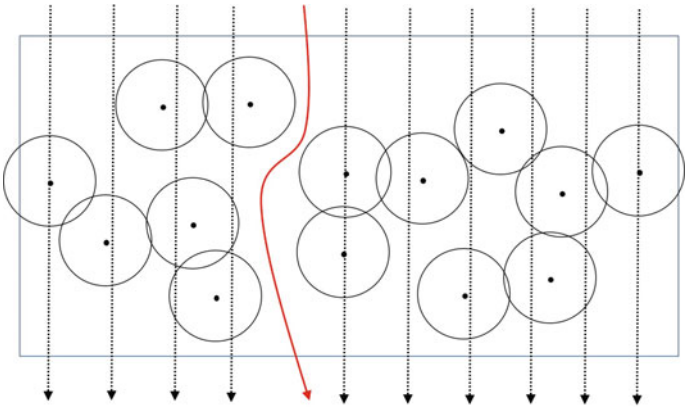


Fig. 3 An example of weak barrier

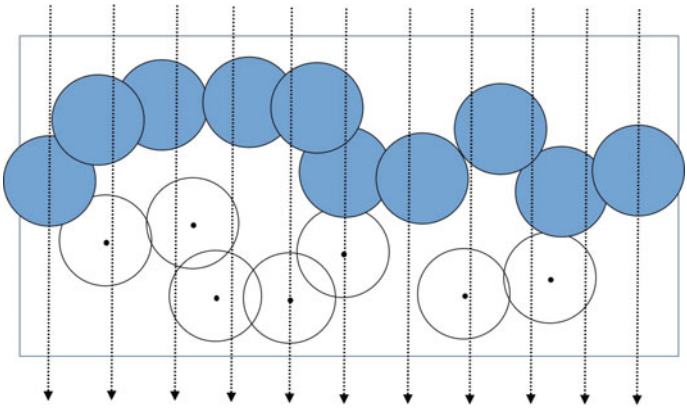


Fig. 4 An example of strong barrier

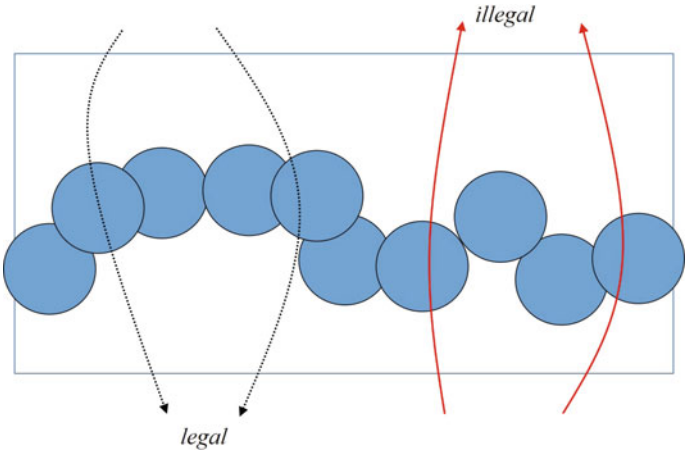


Fig. 5 An example of one-way barrier

with the opposite direction (dotted lines) is considered as *legal* and sensor do not report them.

2.2 Strong k*-Barrier Coverage Using Mobility*

Ban et al. [20] studied how to build strong k-barrier coverage with energy-efficiency by minimizing energy consumption for mobile sensors. They defined two research problems: 1-$BCME$ problem and 1-$GBME$ problem.

2.2.1 1-$BCME$ Problem

Assume that n mobile sensors have initial random locations in a two-dimensional rectangular-shaped area A with a width w and the length l. So, we have a set of sensors, $S = \{s_1, s_2, ..., s_n\}$. The initial position of sensor s_i is represented as (x_i, y_i) and its moved location is also defined as $(\hat{x_i}, \hat{y_i})$. Also, assume that each sensor has an equal sensing range R_s and knows own geographic location information. Note that using the movements of mobile sensors from their initial locations, the objective is to construct barrier coverage in the given area. Because the movement of sensors basically consumes much resource, it is critical how to provide the relocation of sensors with a minimum energy consumption.

Then, 1-$BCME$ (1-Barrier Coverage of Minimum Energy consumption) problem is defined as follows.

Definition 6 (1-$BCME$) Given a rectangular strip A and a set of mobile sensors S, 1-$BCME$ problem is to verify a subset S_c of S and the relocating location $(\hat{x_i}, \hat{y_i})$ for a sensor s_i within S_c such that the total movement distance of every moving sensor is minimized and one sensor barrier within A should be formed by those movements of mobile sensors.

2.2.2 1-$GBME$ Problem

Suppose we divide the area A into N grids where each grid has an equal size. That is, it is defined as $N = n_l \times n_w$, $n_l = \lceil l/2R_s \rceil$, $n_w = \lceil w/2R_s \rceil$. Also, assume that the set of center positions of grids is $G = \{g_1, g_2, ..., g_N\}$, where g_i represents the center position of the ith grid. Since a grid can get 1-barrier coverage if only one sensor is positioned at its center location, we consider that the relocating position of each sensor is chosen from G. Also, assume that if the sensor s_i is moving to the g_j, then the variable x_{ij} is set to 1, and otherwise, x_{ij} is set to 0. The distance between s_i and g_j is defined as d_{ij}. Based on this setting, a sensor barrier is called as a *grid barrier*. In the grid barrier, each sensor is positioned at the center position of the grid and the distance between two neighbor sensors is at most $2 \cdot R_s$. Figure 6 shows an example of *grid barrier*. Within A, a grid is depicted as a small square and a sensor is represented as a small shaded circle. As it can been seen in Fig. 6, we can construct 1-barrier coverage after we form a *grid barrier* by mobile sensors.

Then, we consider the problem how to generate a *grid barrier* with the minimal movement distance. That is, the 1-$GBME$ (1-Grid Barrier Minimum Energy consumption) is defined formally as follows.

Definition 7 (1-$GBME$) Given the set G of central points of grids within a rectangular strip A and a set of mobile sensors S, 1-$GBME$ problem is to decide a subset S_g in S and the destination location p_i from G for each sensor s_i in S_g on condition that that the total movement distance of every moving sensor is minimized and a grid barrier within A should be formed by those movements of mobile sensors.

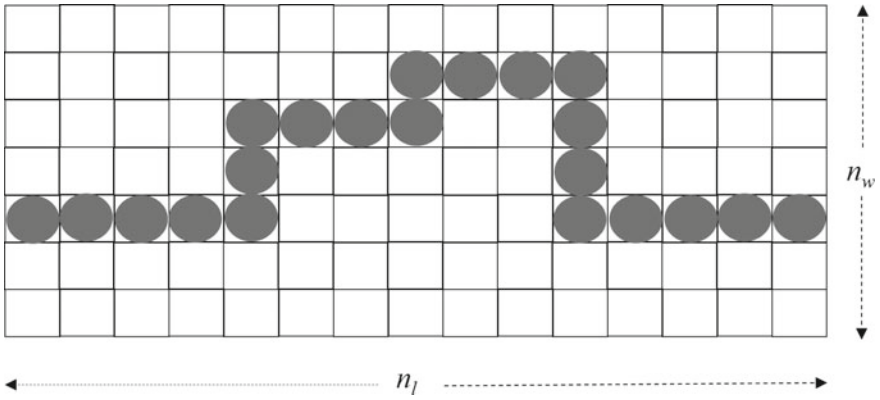


Fig. 6 An example of grid barrier

Ban et al. [20] proved the 1-$GBME$ problem is NP-hard by restrictions to the Knapsack Problem [21].

Theorem 5 1-$GBME$ *problem is NP-hard.*

Proof We first create two restrictions of 1-$GBME$ problem.

Restriction 1: Let $d_{ij} = d_{ik}$, where $\forall j, k \in G, \forall i \in S$. For any sensor s_i, the distance from s_i to any grid is equal.

Restriction 2: The number of mobile sensors consisting of a *grid barrier* is not greater than k.

Then, the movement distance of the sensor is only related to own position by Restriction 1. (e.g., when we choose a sensor, its moving distance is fixed.) It follows that the total movement distance is only related to the set of chosen sensors.

If we assume that c_i is a cost if sensor s_i relocates to g_j, c_i increases as d_i decreases where $c_i = 1/d_i$.

Then, it is proved that 1-$GBME$ problem can be reduced to a new problem which chooses sensors from S with no more than k on condition that the total cost is maximized. So, the new problem also can be formulated as follows.

$$\textbf{Minimize} \sum_{i=1}^{n} c_i \cdot x_i \tag{16}$$

Subject to:

$\sum_{i=1}^{n} 1 \cdot x_i \leq k$

$x_i = \{0, 1\}\ \forall i, i = 1, 2, ..., n$

This formulation is equivalent to Linear Programming Model of Knapsack Problem [21] which is NP-hard. Therefore, we prove 1-$GBME$ problem is NP-hard. □

2.3 Minimizing the Maximum Sensor Movement for Barrier Coverage of a Linear Domain

Chen et al. [22] studied the problem of moving mobile sensors on a line to construct a barrier coverage of a specified segment of the line on condition that the maximum movement distance of the sensor is minimized.

2.3.1 *BCLS* Problem

Assume that $B = [0, L]$ is the barrier which is a line segment from $x = 0$ to $x = L > 0$ on the x-axis. We have the set of mobile sensors, $S = \{s_1, s_2, ..., s_n\}$, which is initially positioned on the x-axis. Also, a sensor s_i ($s_i \in S$) has a sensing range, $r_i > 0$ and each sensor is positioned at the coordinate x_i for the x-axis, where $x_1 \leq x_2 \leq \cdots \leq x_n$. We define that if a sensor s_i is on the location $\hat{x_i}$, then s_i covers the interval $[\hat{x_i} - r_i, \hat{x_i} + r_i]$ that is referred as the *covering interval* of sensor s_i. With these settings, we define $BCLS$ (Barrier Coverage on a Line Segment) problem as follows.

Definition 8 ($BCLS$) Given a set of sensors S and the barrier B, $BCLS$ problem is to identify a set of destinations on the x-axis, $\{y_1, y_2, ..., y_n\}$ for mobile sensors, where $\forall s_i \in S$, s_i moves from x_i to y_i on condition that every location on the barrier B is covered by at least one sensor and the maximum movement distance of mobile sensors is minimized.

So, with the assumption of $2 \cdot \sum_{i=1}^{n} r_i \geq L$, the $BCLS$ problem also can be formulated as follows.

$$\textbf{Minimize} \max_{i=1}^{n} |x_i - y_i| \tag{17}$$

The *decision version* of $BCLS$ problem is to decide if there is a *feasible solution* in which the movement distance of each sensor is at most λ, where $\lambda \geq 0$. Then, according to the ranges of sensors, the *decision version* of $BCLS$ has two cases: *uniform case* and *general case*.

If we consider the range r_i of every sensor is homogenous, then this case is defined as *uniform case*. On the other hand, if the sensors have arbitrary ranges, this case is considered as *general case*.

2.4 Minimizing the Maximum Movement by Mobile Sensors for Barrier Coverage in the Plane

Motivated by Chen et al. [22] to solve the 1-D (line segment) *MinMax* barrier coverage problem, Li et al. [23] introduced the 2-D *MinMax* barrier coverage problem to move mobile sensors in a two-dimensional plane in order to construct a barrier coverage of a specified line segment within the plane.

2.4.1 2-D *MinMax* Problem

Suppose that we have a barrier B which is a line segment on the x-axis in the two-dimensional plane with starting point $[0, 0]$ and ending point $[L, 0]$. Also, assume that we have a set of n mobile sensors $S = \{s_1, s_2, ..., s_n\}$ and the sensors are positioned in the plan with initial positions $p_1, p_2, ..., p_n$, respectively. Also, a sensing range r of every sensor is equal. Then, a sensor s_i has own coordinates of the position p_i, which is denoted by (x_i, y_i). Also, suppose that the barrier line B is covered by all sensor ranges sufficiently. So, we can express it as $\sum_{i=1}^{n} r \geq L$. Note that any sensor in S is movable and is able to relocate itself from the initial location to another targeted location. So, the moving distance of the sensor should be the distance between initial location and the targeted final location of sensor. With these settings, we define 2-D $MinMax$ problem as follows.

Definition 9 (2-D $MinMax$) Given a barrier B and a set of sensors $S = \{s_1, s_2, ..., s_n\}$ with initial positions $p_1, p_2, ..., p_n$, the 2-D $MinMax$ problem is to search for the final positions $\hat{p}_1, \hat{p}_2, ..., \hat{p}_n$ of the mobile sensors in S on the line segment B such that the maximum moving distance of the sensors is minimized.

Then, the 2-D $MinMax$ problem is formulated as follows.

$$\textbf{Minimize} \max_{i=1}^{n} distance(p_i, \hat{p}_i) \tag{18}$$

Also, suppose $R = \sum_{i=1}^{n} r$. If we consider the case $R = L$, then the 2-D $MinMax$ problem is equivalent to a linear bottleneck assignment problem [24]. The problem is to assign n sensors to n grid points on a line segment. If we assume the cost is the moving distance of sensor from initial location to a grid point, an objective of the problem is to search for an optimal assignment to minimize the maximum cost.

2.5 *Distributed Coordination of Mobile Sensors for Barrier Coverage*

Though mobile sensors have the potential to satisfy the coverage requirements, it is a critical issue for a distributed approach to meet these requirements. And it is challenging to meet the required goals for energy consumption, scalability, deployment time as well as to meet dynamic barrier coverage requirements for recovering failed sensors. For those issues, Silvestri et al. [25] proposed *MobiBar* which is an asynchronous and distributed deployment algorithm for k-barrier coverage using mobile sensors.

2.5.1 *MobiBar* Problem

Assume that we have N number of mobile sensors which are deployed over a rectangular area, BoI with the size $L \times W$, where $L \geq W$. Also, assume sensors have

low-cost GPS so that it is possible for sensors to do localization and positioning inaccuracies, bounded by a maximum error ε_{LOC}. Suppose that R_{tx}^{min} and R_{s}^{min} are are the minimum communication and sensing range over BoI, respectively. So, it is set to $\sigma \leq \min\{R_{tx}^{min}, 2R_{s}^{min}\} - 2\varepsilon_{LOC}$ which means that two sensors at a distance σ between them can communicate and their ranges are overlapped despite of positioning and localization errors.

Also, the system assumes that a sensor is able to move in any direction within BoI with a speed of v_{max} m/s. And the system does not consider any special type of sensor (i.e., sink) to coordinate the movements.

Definition 10 ($MobiBar$) Given N number of sensors over a BoI of size $L \times W$ and the consideration of localization inaccuracies, $MobiBar$ problem is to reposition mobile sensors to construct strong k-barrier coverage on condition that every crossing path from a long edge of the BoI to the another side intersects the sensing ranges of at least k distinct sensors.

Silvestri et al. [25] also considers a deployment with the maximum number of barriers, which are parallel to the long edges of the BoI. Then, a lower bound of the maximum number of barriers is calculated as follows.

The maximum number of sensors which can be located at a distance σ over the length L of the BoI is $\lceil \frac{L}{\sigma} + 1$. Hence, a lower bound for the maximum number of barrier B_{max} is calculated as follows.

$$B_{max} = \lfloor \frac{N}{\lceil \frac{L}{\sigma} \rceil + 1} \rfloor \tag{19}$$

2.6 Event-Driven Partial Barrier Coverage

In barrier coverage area, Kim et al. [26] introduced a new framework of barriers, which is referred as *event-driven partial barrier-coverage*. It supports the event-driven environment that has several properties. (1) The set of considered multiple hubs is changed frequently. (2) Whenever a new event occurs or a new hub is added to the system, a formation of barriers can be updated. Then, the *event-driven partial barrier-coverage* guarantees that any moving objects are able to be detected between hubs in the event-driven environment.

2.6.1 System Model

In the system, let us consider a square-shaped area A and a set of mobile sensor nodes S with size n have initial random locations in the area A. After their deployments, a sensor keeps own location to minimize energy consumption. Also, we have a set of interested hub H with size m where a hub is considered as a specific point in the given field. There exists several paths between two hubs. Also, it is possible that the set of

hubs are updated often (i.e., addition of new hubs). In the system, we assume that every sensor has an equal amount of resource as well as has an equal communication radius r.

Define that two sensors s_1 and s_2 are connected with each other and are considered as *neighbors* if the *euclidian distance* between two sensors is at most $2 \cdot r$. Also, given $H_{i,j}$ as a hub pair for the hub h_i and h_j, we define *node-disjoint path* is a sequence of sensor nodes, $s_1, s_2, \ldots, s_k$, between two hubs $h_i, h_j \in H_{i,j}$ with the following requirements.

- s_p and s_{p+1} are connected with neighbors, where $1 \leq p \leq n-1$.
- s_1 is a neighbor of one hub h_i and s_k is a neighbor of another side hub h_j.
- $1 \leq i, j \leq m$ and $1 \leq k \leq n, i \neq j$.

We formally describe the several definition which is used in the system as follows.

Definition 11 (*e-barriers*) Given the found set of paths connected by sensors between each hub pair. e-barrier is a subset of sensors on paths between two hubs h_i, h_j. With the construction of the e-barrier, at least one sensor of e-barrier is able to detect any movements of objects on paths.

Definition 12 (*event-driven partial barriers*) Given the found set of e-barriers for each hub pair in the network, the event-driven partial barrier is a subset of e-barriers. Also, *k-event-driven partial barriers*, called as k-EP barriers, provides a guaranteed detection of mobile objects on paths by at least k sensors for each hub pair.

Note that there may exist several node-disjoint paths for a hub pair $H_{i,j}$. So, we have a set of node-disjoint paths $P_{H_{i,j}} = p_1, p_2, ..., p_a$ between a hub pair $H_{i,j}$, where $a > 0$. Moreover, by Definition 11, multiple e-barriers can be formed between two hubs on condition that each e-barrier is independent, which means there are no common sensors with other e-barriers between the hubs h_i, h_j. That is, we have a set of e-barriers, $E_{i,j} = e_1, e_2, ..., e_b$, where $b > 0$. And each e-barrier consists of a sequence of connected sensors, $s_1, s_2, \ldots, s_a$, where $s_1 \in p_1, s_2 \in p_2, ..., s_a \in p_a$.

2.6.2 *MinSkEP* Problem

Basically, the term *partial* in *event-driven partial barrier* in Definition 12 implies *reducible* with dependency in the system. Because some sensors are common member among e-barriers at different hub pairs, one critical issue of *event-driven partial barrier* indicates how to construct *k-event-driven partial barriers* (k-EP barriers) efficiently using the property of dependency among e-barriers.

Based on the above issue, our problem is defined formally as follows.

Definition 13 ($MinSkEP$) Given a set of wireless sensor nodes S and a set of hubs H deployed over an square-shaped area A, *the minimum number of sensors for k-EP barriers ($MinSkEP$)* problem is to minimize the number of sensors so as to form k-EP barriers among given hub pairs.

Figure 7 shows initial status of the proposed system. Suppose that a set of sensors are randomly deployed in the two-dimensional area A, which a sensor is represented as a small circle. Also, we suppose a set of hubs $H = \{h_1, h_2, h_3, h_4, h_5, h_6\}$, and then, we have three hub pairs such as $H_{1,2} = \{h_1, h_2\}$, $H_{3,4} = \{h_3, h_4\}$, $H_{5,6} = \{h_5, h_6\}$. In Fig. 7, a hub is depicted as a small triangle. Then, suppose that we have found node-disjoint paths (or independent paths) between each hub pair, which each path is described as a dotted line between two hubs. In Fig. 7, we have $P_{H_{1,2}} = p_1, p_2, p_3, p_4$, $P_{H_{3,4}} = p_5, p_6, p_7$, $P_{H_{5,6}} = p_8, p_9, p_{10}$, respectively.

Also, Fig. 8 is an example of the constructed k-EP barriers with $k = 3$. With the created k-EP barriers with $k = 3$, we guarantee that at least k (=3) sensors can detect any movement of objects among hub pairs. To solve the $MinSkEP$ problem, we choose the set of e-barriers: e_1, e_2, e_3 for $H_{1,2}$, e_4, e_5, e_6 for $H_{3,4}$ and e_7, e_8, e_9 for $H_{5,6}$, which each e-barrier is depicted as a solid line in Fig. 8. Remind that we should use the possible minimum number of sensors to solve $MinSkEP$ problem. In Fig. 8, we have used 25 sensor nodes to construct k-EP barriers with $k = 3$.

2.6.3 Construction of Event-Driven Partial Barriers

For an initial setup, we perform the following steps to search for possible node-disjoint paths $P_{H_{i,j}}$ for hub pairs.

Step 1.
A *flow graph* $\mathscr{G} = (V(\mathscr{G}), E(\mathscr{G}))$ is generated as follows.

- For every sensor $s \in S$, put two vertices s_{in} and s_{out} into $V(\mathscr{G})$.
- For each sensor $s \in S$, include a directed edge $\overrightarrow{s_{in}, s_{out}}$ to $E(\mathscr{G})$.

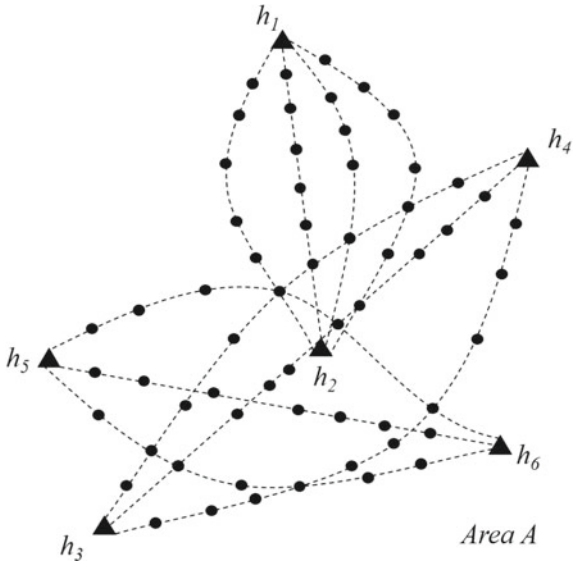


Fig. 7 Initial status with a set of sensors, a set of hubs, and a set of node-disjoint path between hubs

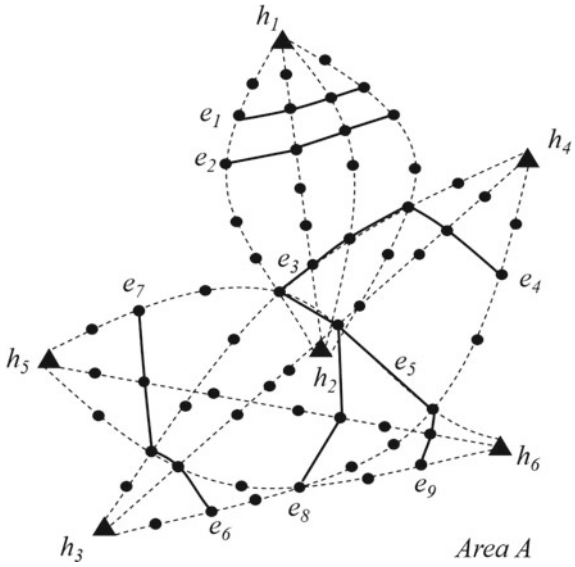


Fig. 8 Construction of k-EP barriers with $k = 3$

- For every pair of sensor nodes s and t in S that are neighbors, add two directed edges $\overrightarrow{s_{out}, u_{in}}$, $\overrightarrow{u_{out}, s_{in}}$ into $E(\mathscr{G})$.
- Add hubs h_i and h_j to $V(\mathscr{G})$.
- Add an edge (h_i, s_{in}) to $V(\mathscr{G})$ for a neighbor sensor s of h_i
- Also, consider to add an edge (s_{out}, h_j) to $V(\mathscr{G})$ for a neighbor sensor s of h_j.

Step 2.
Assign a capacity of 1 to each edge in the flow graph $\mathscr{G} = (V(\mathscr{G}), E(\mathscr{G}))$. Given a source h_i and a destination h_j, implement a maximum flow algorithm, such as Edmonds–Karp algorithm [27]. We note that a flow is equal to a node-disjoint path from h_i to h_j. So, we finally get the set of node-disjoint paths $P_{H_{i,j}}$ for every hub pair. Then, we find a set of e-barriers, $E_{i,j} = e_1, e_2, ..., e_b$, where $b > 0$. Each e-barrier is independent and it is created by a sequence of sensor nodes on the found $P_{H_{i,j}}$. For example, the sensors in e_1 cannot be used at e_2. For every hub pair $h_i, h_j \in H$, we implement the Step 1 and 2. It follows that we can calculate a different set of e-barriers $E_{i,j}$ for every hub pair.
Step 3.
For each sensor, calculate its frequency which is the number of times used in constructed e-barriers of different hub pairs.

Then, the pseudocode by Step 1, 2, 3 is represented in Algorithm 1 which is named as *Initial-Setup*.

In order to solve $MinSkEP$ problem, Kim et al. [26] introduced two approaches: *Greedy-Shared-Barrier* and *Greedy-Shared-Sensor*. Both approaches returns EPS which is the number of sensors of consisting of complete k-EP barriers.

Algorithm 1 ***Initial-Setup***
Inputs: S, H, r, k

search for a set of node-disjoint path, $P_{H_{i,j}}$, for each hub pair $H_{i,j}$;
calculate a set of e-barrier, $E_{i,j}$, for each $H_{i,j}$;
set a frequency of each sensor shared by different e-barriers: $f_i = 0$;
for each sensor s_i, identify if s_i is utilized as e-barrier in a different hub pair;
if there exists **then**
    increase a frequency of s_i as f_i++;
end if

For *Greedy-Shared-Barrier*, the above steps are iterated until k-EP barriers are generated.

- Verify e-barrier e_{max} with the maximum number of shared sensors by other e-barriers.
- Identify e-barriers affected by sensors e_{max}.
- Activate the identified e-barriers as k-EP barriers.
- Search for a sensor set S_{act} in the activated e-barriers and then calculate the size of S_{act} as $|S_{act}|$.
- Add $|S_{act}|$ into EPS.

Also, the pseudocode is described in Algorithm 2.

Algorithm 2 ***Greedy-Shared-Barrier***
Inputs: S, H, r, k, Output: EPS

set $EPS = 0$;
call Algorithm 1 for Initial-Setup;
while k-EP barriers are not built **do**
    set a set of current activated e-barriers: $E_{act} = \emptyset$;
    set a sensor set which is affected by E_{act}: $S_{act} = \emptyset$;
    choose e_{max} with the largest number of shared sensors by other e-barriers;
    activate e_{max} as well as e-barriers which are affected by sensors in e_{max};
    add e_{max} and the activated e-barriers to E_{act};
    update E_{act} to k-EP barriers;
     verify S_{act} that is covered by E_{act};
     update $EPS = EPS + |S_{act}|$;
     if k-EP barriers are constructed **then**
        break;
     end if
end while
return EPS

On the other hand, for *Greedy-Shared-Sensor*, the above steps are repeated until k-EP barriers are constructed.

- Search for a sensor s_{max} with the largest frequency.
- Verify e-barriers covering s_{max} and then update those e-barriers as k-EP barriers.

- Identify a sensor set S_{act} affected by the updated e-barriers and then calculate $|S_{act}|$.
- Increase EPS by adding $|S_{act}|$.

Then, the pseudocode of *Greedy-Shared-Sensor* is described in Algorithm 3.

Algorithm 3 ***Greedy-Shared-Sensor***
Inputs: S, H, r, k, Output: EPS

```
1: set EPS = 0;
2: call Algorithm 1 for Initial-Setup;
3: while k-EP barriers are not formed do
4:    set E_act = ∅;
5:    set S_act = ∅;
6:    identify a sensor s_max with the largest frequency f_max;
7:    active e-barriers affected by s_max and add them to E_act;
8:    update E_act to k-EP barriers;
9:    find S_act which is covered by E_act;
10:   verify S_act that is included in E_act;
11:   update EPS = EPS + |S_act|;
12:   if k-EP barriers are constructed then
13:      break;
14:   end if
15: end while
16: return EPS
```

2.7 *Resilient Event-Driven Partial Barrier Coverage Using Mobile Sensors*

After k-EP barriers in mobile sensor networks are constructed, there may exist sensor failures due to energy exhaustion of sensors. So, the management of those failures should be considered to support the system continuously. Because of dependency property of k-EP barriers, a failure of some sensor in k-EP barriers may cause not only a crack of k-EP barriers but also a demolition of the whole system for event-driven barrier coverage. Kim et al. [28] deal with this critical issue, and then, we review which problem is addressed formally and which strategy can be applied.

2.7.1 Resilient Event-Driven Partial Barriers

To handle the sensor failures in k-EP barrier, we simply may consider additional deployments of mobile sensors into the network. But, those simple addition of sensors will increase the cost of maintenance. Therefore, it is highly appropriate to use the current deployed mobile sensors in the field in order to solve the failures of sensors.

It follows that the movements by existing mobile sensors are utilized to recover both the sensor failures and the collapsed k-EP barriers without any addition of sensors.

Now, we formally define a *resilient event-driven partial barriers*.

Definition 14 (*resilient event-driven partial carriers*) Given a set of mobile sensors S and a set of hubs H in a square-shaped area A, assume that k-EP barriers have been constructed initially. When initial k-EP barriers are collapsed because of sensor failures, *resilient event-driven partial barriers* are to remedy those failures by mobile sensors and to keep complete k-EP barriers continuously.

2.7.2 *MinTMove* Problem

Note that the mobility of sensors are used to perform various objectives. But, the energy of mobile sensors is basically limited and also the movement operation of sensors consumes much more energy than other operations by sensors (i.e., computation) [29]. Also, since it is difficult for mobile sensor to be recharged after their initial deployment, minimizing movement of mobile sensors should be an important issue in wireless sensor networks [30–33].

Similarly, the minimum movement of sensors are desirable to support *resilient event-driven partial barriers*. Then, we define this problem formally as follows.

Definition 15 ($MinTMove$) Given the collapsed initial k-EP barriers by sensor failures, $MinTMove$ problem is to minimize a total movement of mobile sensors so as to keep complete k-EP barriers continuously using mobility of sensors without any additional deployment of sensors.

It has been known that a difficulty of *MinTMove* problem to provide an efficient fault management is a dependency property of *event-driven partial barriers*. We simply may not apply a strategy that we choose the sensor with a minimal distance from the failure sensor and locate the sensor to the position of failed sensor because we may have additional serious problem such as an occurrence of another lost connection within current k-EP barriers by the movement. Furthermore, we should keep current network status equally when compared with initial network status, i.e., the same number of node-disjoint paths. Hence, the moving strategy including the selection of moving sensors should be considered carefully to solve $MinTMove$ problem and to maximize the network lifetime.

Now, we show several scenarios or cases of moving sensor selections in Figs. 9, 10, 11, and 12. At those figures, suppose that a sensor is represented as a small circle, a hub is depicted as a small triangle. Also, assume that a node-disjoint path between two hubs is described as a dotted line and the constructed k-EP barriers with $k = 3$ are represented as solid lines. Then, we have two hub pairs: $H_{1,2} = \{h_1, h_2\}$, $H_{3,4} = \{h_3, h_4\}$. As k-EP barriers with $k = 3$, there exist e_1, e_2, e_3 between h_1 and h_2 and e_4, e_5, e_6 between h_3 and h_4.

Figure 9 shows an occurrence of sensor failure. Suppose a sensor f in e_5 is failed due to its energy depletion. Such a failure of sensor f will cause a demolition of

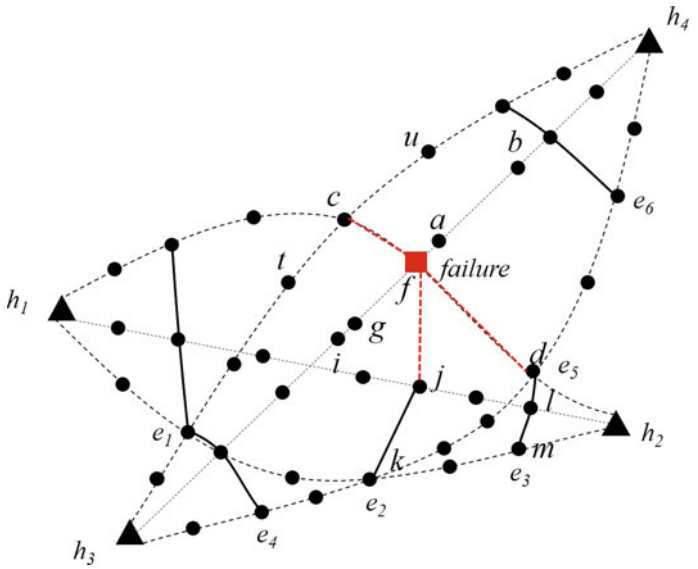


Fig. 9 Occurrence of failure

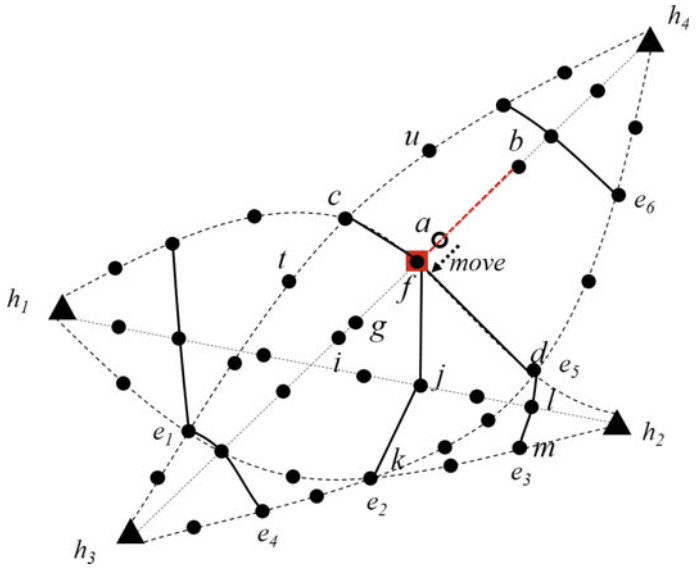


Fig. 10 Possible sensor movement case 1

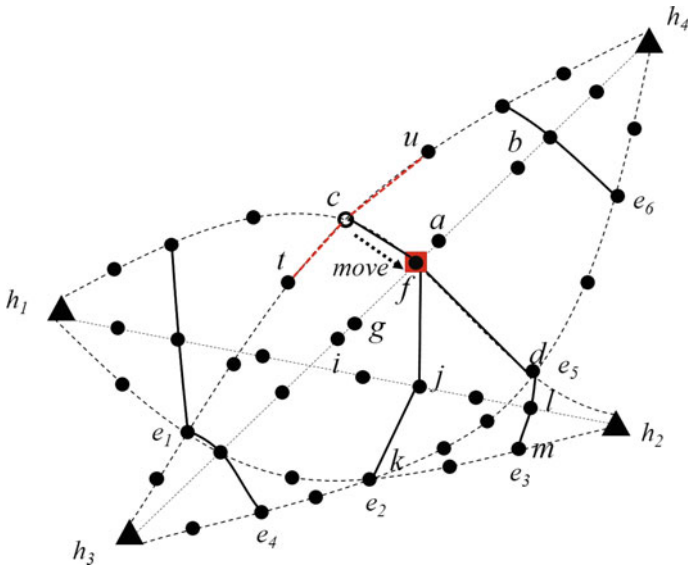


Fig. 11 Possible sensor movement case 2

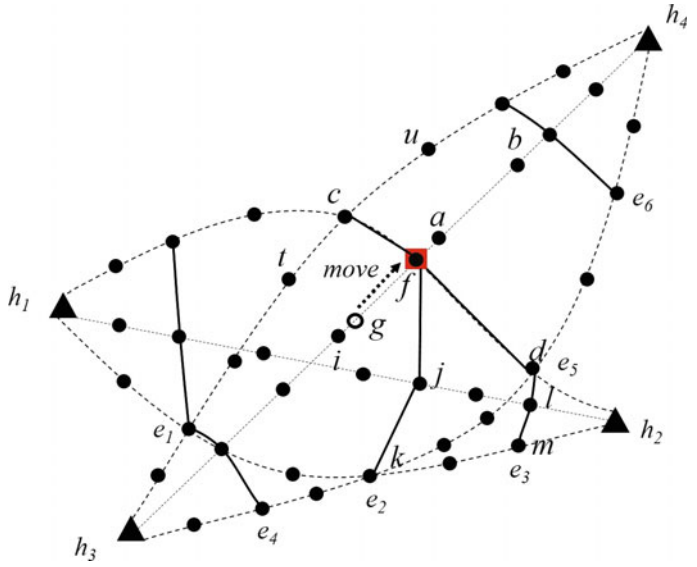


Fig. 12 Possible sensor movement case 3

k-EP barriers by lost connections with sensor c, d, j. That is, we have lost edges $\overline{f, c}$, $\overline{f, d}$, $\overline{f, j}$ because of the failure in the network.

To remedy the sensor failure, the sensor a with the minimum distance for f can move to the position f and then be replaced as Fig. 10. This movement allows us to recover the failure f. However, we may have an additional lost edge if the moved position is out of communication range of sensor b. (i.e., the *euclidian distance* between the moved position and b is greater than $2 \cdot r$). The lost edge will also cause the collapse of one node-disjoint path between h_3 and h_4.

Figure 11 describes the selection of next closest sensor c which is a part of another node-disjoint path between h_3 and h_4. If a sensor c moves to the position of failure sensor as Fig. 11, the failure will be recovered. However, such a movement of c may result in additional lost connections with sensor t and u if the moved position is out of communication range with t and u. What is worse, the node-disjoint path will not be maintained if t is out of communication range with u.

Then, as Fig. 12, we may select the next possible closest sensor g, which is located on the equal node-disjoint path with the failure sensor f. After a sensor f is replaced by g, we may have lost connection if the new position of g is still within the communication range of sensor i. If so, this case should provide the case of the appropriate movement selection.

2.7.3 Construction of Resilient Event-Driven Partial Barriers Using Movements of Mobile Sensors

Because we simply cannot choose the closest sensor to replace the failed sensor due to the dependency property of k-EP barriers, the moving strategy should be considered carefully to solve $MinTMove$ problem. Kim et al. [28] proposed a heuristic approach, referred as *Greedy-Point-Movement*, to decide moving sensors to recover from failures. Before implementing *Greedy-Point-Movement* approach, it requires we need to identify the set of failed sensors, F by checking the all deployed sensors where the number of sensors is n. The *Greedy-Point-Movement* has the following steps.

- Select a failed sensor s_f from the set of failed sensors, F.
- Search for the closest node with s_f.
- Case 1: If the closest sensor s_{c_1} is available on a hub path and is not a part of another barrier within k-EP barriers, then verify if we have any loss of node-disjoint path between hubs when the s_{c_1} moves into a position of s_f.
  - If there exists any lost connection, go to Case 2.
  - If not, move the sensor s_{c_1} to the position of s_f and add the movement distance m to a total moving distance $totalmove$.
- Case 2: If s_{c_1} is a part of of another barrier within current k-EP barriers, then search for the next closest sensor s_{c_2} on the node-disjoint hub path. Also, we verify if we have any loss of node-disjoint path when s_{c_2} moves to s_{c_1}.

  - If there exists any lost connection, go to Case 3.
  - If not, move the sensor s_{c_2} to the position of s_{c_1} and also move s_{c_1} to the position of s_f. And add the movement distance by both movements to a total moving distance $totalmove$.

- Case 3: If current closest sensor s_{c_2} for s_f is not satisfied by Case 1 and 2, find the next closest sensor s_{c_3} of s_f and verify if the sensor is satisfied by Case 1 or Case 2.
- If all failed sensors are recovered and k-EP barriers are constructed, return $totalmove$. Otherwise, return $failure$.

Then, we present the pseudocode of *Greedy-Point-Movement* in Algorithm 4.

Algorithm 4 ***Greedy-Point-Movement***

Inputs: S, H, F, r, k, n, Output: $totalmove$ or $failure$

```
1: set totalmove = 0;
2: while k-EP barriers are not formed do
3:    choose a failure sensor s_f from F;
4:    set current closest rank i = 1;
5:    set m = 0;
6:    while s_f is not recovered do
7:       search for the closest sensor s_ci from s_f;
8:       if s_ci is not exploited at another e-barrier then
9:          verify if there exists any lost connection when s_ci moves to a position of s_f;
10:         if there is no lost connection then
11:            move s_ci to the position of s_f;
12:            add m to totalmove;
13:            break;
14:         end if
15:      else if s_ci is a part of another e-barrier then
16:         i++;
17:         search for the next closest sensor s_ci from s_f;
18:         verify if there exists any lost connection when s_ci moves to a position of s_f;
19:         if there is no lost connection then
20:            move s_ci to the position of s_f;
21:            add m to totalmove;
22:            break;
23:         end if
24:      else
25:         i++;
26:      end if
27:      if we cannot find s_ci to replace with s_f without lost connection then
28:         return failure
29:      end if
30:   end while
31: end while
32: return totalmove
```

2.8 *Barrier Coverage of Variable Bounded-Range Line-of-Sight Guards*

Though many studies focused on barrier coverage of the square-shaped region, the barrier coverage in various shaped areas also has been studied. Kloder et al. [34] formalized the problem of barrier which is to prevent undetected intrusion in a particular region using mobile entity such as mobile sensors, robots.

2.8.1 Problem Statement

Let us consider a mobile object which is called as *intruder*. The intruder can be located at some point inside a start region S_1 and may try to enter some stop region S_2. A barrier can be defined as the set of sensors (or robots) formation which allows us to guarantee the penetration of intruders from entering S_2 should be detected by at least one sensor. Kloder et al. [34] defines these robots or sensors as *guards*. Simply, they also can be considered as *fences*.

Suppose that a point of intruder $(x_I, y_I) \in \mathrm{R}^2$ and the intruder can only be in the obstacle-free space $\mathtt{W} \subset \mathrm{R}^2$. Also, assume that the intruder is located in the start set $S_1 \subset \mathtt{W}$ and can move to some location in the stop set $S_2 \subset \mathtt{W}$, where S_1 and S_2 are bounded by polygons. Also, we assume that *guards* can see a straight line until they go outside of ranges. Figure 13 describes a sample of barrier problem domain.

Then, a guard g_j can be defined as $q_j = (x_j, y_j, \boldsymbol{\theta}_j, r_j) \in \mathrm{R}^2 \times S^1 \times \mathrm{R}^+$. The guard is positioned at (x_j, y_j) which is with direction $\boldsymbol{\theta}_j$ up to a distance of its range r_i. And suppose that guards are not able to see through walls. In addition, for a guard $q = (x, y, \boldsymbol{\theta}, r)$, we define that a $visibilityregion\, V(q)$ is the set of points

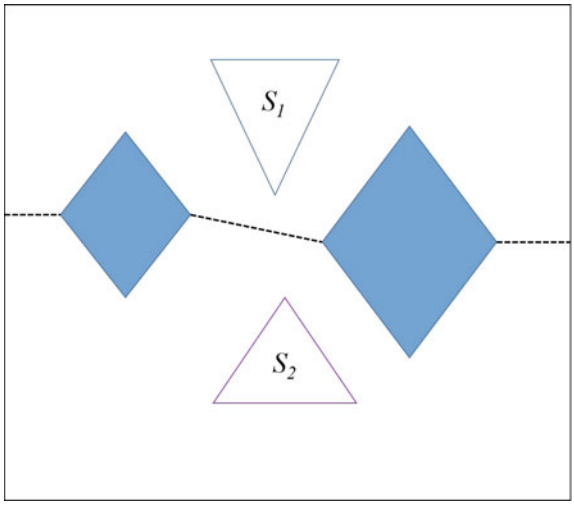


Fig. 13 Sample of barrier problem domain. The shaded areas represent obstacles

which q can check. Because a guard q has a straight-line view, $V(q)$ should be the maximum connected component of $\{x + kcos\boldsymbol{\theta}, y + ksin\boldsymbol{\theta})|0 \leq k \leq r\} \cap \mathbb{W}$ which includes (x, y). So, these guards are called as *segment guards*.

Then, we can define that a set of guards $\{q_1, q_2, ..., q_n\}$ is a *barrier* if and only if every path from S_1 to S_2 in $\mathbb{W}$ intersects $V(q_j)$ for at least one j. It follows that $\{q_1, q_2, ..., q_n\}$ is a *barrier* if and only if S_1 and S_2 are within separate connected components of $\mathbb{W} - \bigsqcup_{j=1}^{n} V(q_j)$.

With these settings, our objective is to search for the minimum length barrier. That is, for a set of guards $\{q_1, q_2, ..., q_n\}$, the objective of the problem can be formulated as follows.

$$\textbf{Minimize} \sum_{j=1}^{n} r_j \tag{20}$$

2.9 Barrier Coverage by Mobile Sensors of Polygon Region

Bhattacharya et al. [35] consider a barrier by mobile sensors in the planar region. It follows that they considered how to move mobile sensors to the perimeter of a region delimited by a polygon.

2.9.1 *MinSum* Problem on a Simple Polygon

Given a geometric planar region, let us assume that each sensor identifies the region to be barrier covered and sensors are able to move autonomously in the region. Then, sensors can move from their starting locations to the new locations on the perimeter of the polygon. It follows that we should decide how to move those sensors within the polygon region in order to minimize a total movement distances of mobile sensors. W call this as the *MinSum* barrier coverage problem. Figure 14 shows the movement issue in polygon region, where a sensor is depicted as small circle. The left side of Fig. 14 shows initial locations of sensors in the polygon region and the right side of Fig. 14 represents the status that sensors move from initial positions to the specific perimeter locations to repair the possible security hole and prevent the penetrations of intruders into the polygon.

Suppose P be a simple polygon and PB be the boundary of P. Also, assume that PB is oriented in the clockwise direction. Let $P_1, P_2, ..., P_m$ be the vertices of P ordered in the positive direction and suppose the edges of P be $e_1, e_2, ..., e_m$, where e_i has endpoints P_i and P_{i+1}, $1 \leq i \leq m$. Also, we consider n mobile sensors that are positioned in the interior or on the perimeter of P. And assume that each sensor knows own geometric coordinates and the simple polygon. In more detail, n sensors are located at locations $L_1, L_2, ..., L_n$. Let $\hat{L}_i$ be the destination location of L_i on PB, where $i = 1, 2, ..., n$. In the problem, we consider mobile sensors have the only

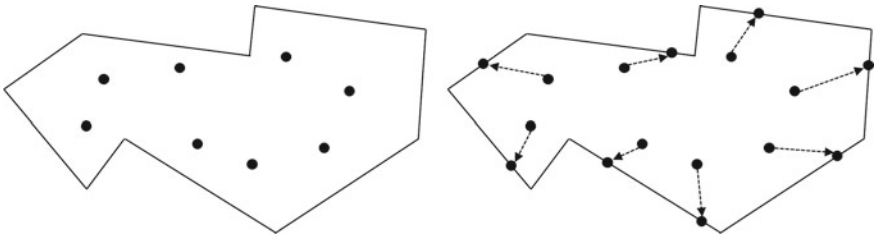


Fig. 14 Sensor's movement from initial location to the location on the perimeter of a polygon region

straight-line movement trajectory and mobile sensors have unrestricted movements (i.e., it is possible to move sensors outside the boundary of the polygon).

Then, for $\forall i$, $i = 1, 2, ..., n$, the objective of the problem can be formulated as follows.

$$\textbf{Minimize} \sum_{i=1}^{n} distance(L_i, \hat{L}_i) \tag{21}$$

3 Sweep Coverage Using Mobility

In this section, we study *sweep coverage* based on mobility of sensors. The concept of *sweep coverage* was introduced by Gage [6] firstly. Then, Cheng et al. [36, 37] formally defined the system model and problems of *sweep coverage*. Different from previous static coverage, the *sweep coverage* only monitors specific points of interests (PoI) periodically and the coverage at each PoI should be time-variant.

3.1 Sweep Coverage in Sensor Networks

Some applications may have dynamic requirements of the time dimension. For the application of patrol inspection, instead of monitoring an entire area at any moment, specific points of interest (PoI) can be monitored periodically, which is defined as *sweep coverage* introduced by Cheng et al. [36, 37]. Different from static coverage, *sweep coverage* considers that each PoI is time-variant with a guaranteed coverage interval. Hence, simply apply traditional scheme of static coverage is not appropriate.

Figures 15, 16, and 17 show an example of *sweep coverage* with PoI in area A. Within those figures, a point of interest is depicted as a small circle. Figure 15 describes a set of PoI. So, we have 14 points from a to n within area A. Then, Fig. 16 shows the *sweep coverage* with one mobile sensor. A trajectory of the sensor is represented as a directed dotted edge. So, for *sweep coverage*, the mobile sensor is able

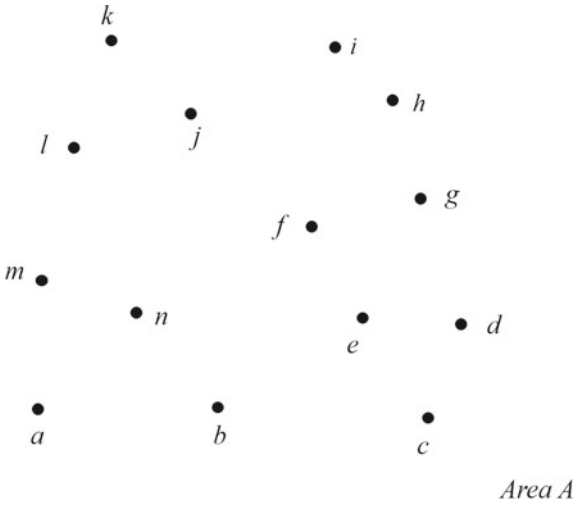


Fig. 15 An example of initial PoI

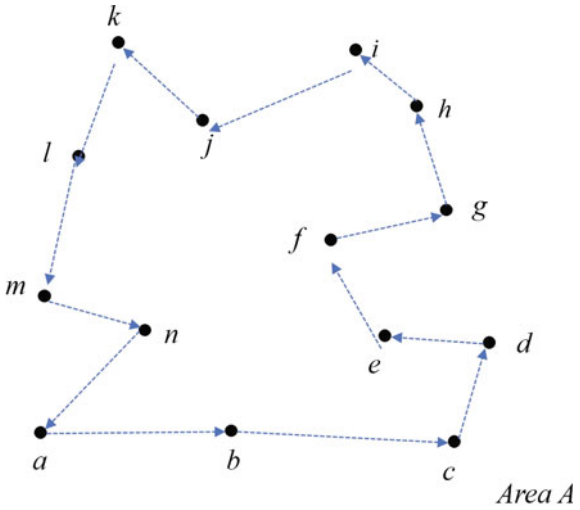


Fig. 16 An example of sweep coverage with one mobile sensor

to visit to PoI with the order of $\{a, b, c, ..., m, n, a\}$. Also, Fig. 17 explains the case of *sweep coverage* with four mobile sensors. As is is seen in Fig. 17, we consider four subareas such as A-1, A-2, A-3, A-4, respectively. Each subarea is monitored by one mobile sensor. Then, for A-1, the visiting order by a mobile sensor is $\{a, b, m, n, a\}$. For A-2, a visiting trajectory of a mobile sensor will will be $\{c, d, e, c\}$. For A-3, a visiting trajectory should be $\{f, g, h, i, f\}$. Also, a mobile sensor will do own movement as $\{j, k, l, j\}$ to perform *sweep coverage*.

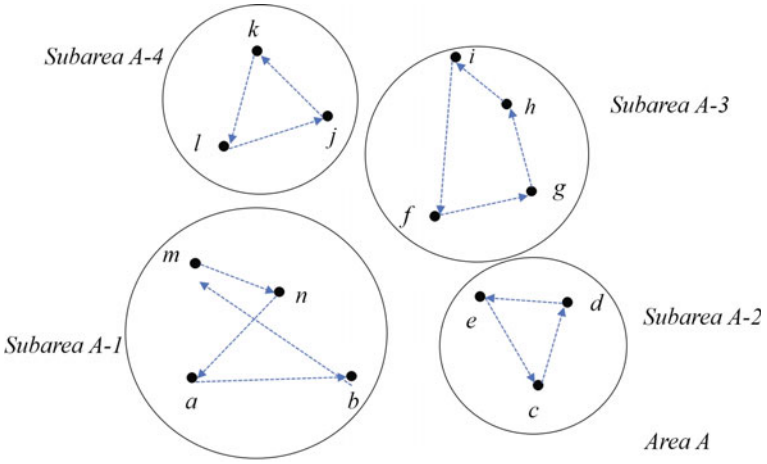


Fig. 17 An example of sweep coverage with four mobile sensors

3.1.1 *Min-sensor Sweep-Coverage* Problem

Let us assume that we have a set of mobile sensors, $S = \{s_1, s_2, ..., s_n\}$. Those mobile sensors are randomly or strategically used to cover m points of interests (PoI), $H = \{h_1, h_2, ..., h_m\}$, in the area. Also let us assume that $d_{i,j}$ is the Euclidean distance between PoI h_i and h_j and every sensor has the equal moving speed v. And, at a specific time moment, a PoI is visited by a sensor if and only if the sensor is located at the PoI. In *sweep coverage*, it is required that the PoI should be monitored at least once with a specific time interval to ensure that we provide an event detection in a certain delay bound. Then, based on these settings, three important definitions, which are referred as t-sweep coverage, coverage scheme α and $global\ t$-sweep coverage, are represented as follows.

Definition 16 (*t-sweep coverage*) Given a set of sensors S and a set of PoI H, a PoI is to be t-sweep covered by a coverage scheme α if and only if it is monitored at least once with every t time intervals by mobile sensors performed by α.

Definition 17 (*coverage scheme α*) A coverage scheme α is a schedule of the mobile sensor movement. If a PoI is t-sweep covered, then the time unit r is defined as the *sweep period* of the PoI. Also, it is possible different PoI can follow different *sweep periods*. So, the PoI h_i should be monitored once at every t_i time unit.

Definition 18 (*$global\ t$-sweep coverage*) A set of PoI H is to be globally t-sweep covered by a coverage scheme α if and only if every PoI h_i in H is t_i-sweep covered by α.

Note that if $t_i = t$ for every PoI, it is said to be a simplified problem referred as $global\ t$-sweep coverage. Based on this definition, we formally define $min\text{-}sensor\ sweep\text{-}coverage$ problem as follows.

Definition 19 (*min-sensor sweep-coverage problem*) Given a set of sensors S and a set of PoI H, the *min-sensor sweep-coverage* problem is to decide the minimum number of mobile sensors to meet the desired *global* sweep coverage with the constraints of t_i-sweep coverage for each PoI.

Based on the Definition of *global* t-sweep coverage, Du et al. [38] also formally defined the problem for *global* t-sweep coverage with minimum mobile sensors. Given the graph $G(V, E)$ where V is the vertex set with m PoI and E is the edge set with the straight lines between every pair of PoI. Also, consider that the straight line between PoI h_i and h_j has the length $d_{ij} \cdot T_i$ as a weight, where T_i is a starting time of measurement and t is the sweep period of PoI. Then, for $\forall[T_i, T_i + t]$, the integer variable X_{ijs} can be defined as follows.

$$X_{ijs} = \begin{cases} 1, \text{ if sensor } s \text{ passes} E_{ij} \\ 0, \text{ otherwise.} \end{cases}$$

We formally define the *global* t-sweep coverage with minimum mobile sensors as follows.

$$\textbf{Minimize } m \tag{22}$$

Subject to:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} X_{ijs} |d_{ij}| \leq L = vt, (\forall[T_i, T_i + t], s = 1, 2, 3, ..., m) \tag{23}$$

$$\sum_{i=1}^{n} X_{ilp} \geq 1, (\exists p \in \{1, 2, 3, ..., m\}, l = 1, 2, 3, ..., n) \tag{24}$$

Equation (23) means that the path trajectory of mobile sensors is no greater than the length L that is the maximum trajectory by the mobile sensor within t sweep interval. And the total trajectory should be $L = v \cdot t$. Equation (24) restricts that every PoI should be visited by at least one of m sensors during every sweep interval.

3.1.2 Hardness of *Min-sensor Sweep-Coverage* Problem

Cheng et al. [36, 37] proved the *min-sensor sweep-coverage* problem is NP-hard by showing a reduction from the Traveling Salesman Problem (TSP), which is described at the following Theorem.

Theorem 6 *Given a set of sensors S, a set of PoI H and the sweep coverage time period requirement of PoI, min-sensor sweep-coverage problem is NP-hard.*

We reduce the TSP problem to the *min-sensor sweep-coverage* problem so as to prove the its NP hardness as follows.

Proof Given a set of m sites $U = \{u_1, u_2, ..., u_m\}$ in the two-dimensional plane. The TSP problem's objective is to find the shortest trajectory to visit every site once and return to the starting location. The decision problem for TSP problem is checking if there exists a cycle with a length which is not greater than given value L. Then, if the given TSP problem (U, L) has a solution, $\frac{L}{v}$-sweep coverage can be supported by one sensor. That is, all sites can be visited by a sensor at least once with every $\frac{L}{v}$ time interval. If the $min\text{-}sensor\ sweep\text{-}coverage$ problem has a solution by one sensor, the decision problem of TSP also has a solution. By any interval of $t = \frac{L}{v}$ time interval, every site should be visited by the sensor at least once by the coverage scheme α. It follows that the coverage scheme α supports a trajectory on condition that every site is visited at least once. Hence, the total distance of the trajectory is at most $\frac{L}{v} \cdot v = L$. By the reduction, we prove that the $min\text{-}sensor\ sweep\text{-}coverage$ problem is NP-hard. □

Theorem 7 *Given a set of sensors S, a set of PoI H and the sweep coverage time period requirement of PoI, $min\text{-}sensor\ sweep\text{-}coverage$ problem cannot be approximated with a factor of 2 unless $P = NP$.*

Proof To show that the problem cannot have any polynomial time algorithm with a approximation ratio $\leq 2 - \varepsilon$ where $\varepsilon > 0$ unless $P = NP$, let us assume, by contradiction, that there exists the polynomial time approximation algorithm referred as *APPR*. Also, let the decision problem of TSP with L be the distance of the optimal trajectory of TSP. Then, the corresponding $min\text{-}sensor\ sweep\text{-}coverage$ problem still has an optimal solution by one mobile sensor. And the number of sensors implemented by *APPR* should be at most $(2 - \varepsilon) \cdot 1$. It follows that the optimal solution should be 1 and this solution can be performed within polynomial time. So, this implies that the original TSP has a solution. □

3.2 Area Sweep Coverage

Different from previous *sweep coverage* problem (or *point sweep coverage*, Gorain and Mandal [40] introduced the concept of *area sweep coverage* that every point of a bounded area should be covered at least once in a specified time interval. So, same strategy used in *point sweep coverage* should not be utilized at *area sweep coverage* because we have infinite number of points in the bounded area.

3.2.1 *Area Sweep Coverage* Problem

Suppose A is a given bounded area of interest (AoI) and we have a set of mobile sensors $S = \{s_1, s_2, ..., s_m\}$. The area A is said to be t-area sweep covered if and only if every point of A is within the sensing range of at least one mobile sensor in every t time interval, where t is a sweep period of the A. Then, we formally defined the *area sweep coverage* problem as follows.

Definition 20 (*area sweep coverage*) The *area sweep coverage* problem is to search for the minimum number of mobile sensors on condition that every point of the given area A is t-sweep covered for a given time interval t.

Now, we show the *area sweep coverage* problem is NP-complete in [40].

Theorem 8 *The area sweep coverage problem is NP-complete.*

Proof Given a set of mobile sensors with their corresponding path, it is possible to check if every point of the AoI is covered at least once by a mobile sensor within polynomial time. Therefore, the decision version of the *area sweep coverage* problem is in NP. Li et al. [41] has shown that the problem of covering a bounded area with the minimal number of static sensor is NP-hard. It follows that covering a bounded area is a special case of the *area sweep coverage* problem if the sweep interval $t = 0$. So, the decision version the *area sweep coverage* problem should be NP-hard. Hence, we prove that the *area sweep coverage* problem is NP-complete. □

3.3 Energy Efficient Sweep Coverage

For *sweep coverage*, Gorain and Mandal [42] have used both static and mobile sensors instead of using only mobile sensors. They have shown that the combination of static and mobile sensors is more efficient when we consider the total number of sensors for *sweep coverage* problem. With the use of static and mobile sensors, Gorain and Mandal [42] introduced a concept of t-$GSweep$ coverage, which is a variation of *sweep coverage* problem.

Also, Gorain and Mandal [40] introduced an energy efficient sweep coverage problem referred as *EEGSweep coverage* problem based on the t-$GSweep$ coverage. For a given set of point, the goal of *EEGSweep coverage* problem is to minimize energy consumption by a combined set of static and/or mobile sensors with guaranteed *sweep coverage*.

3.3.1 *EEGSweep Coverage* Problem

Firstly, we define t-$GSweep$ coverage as follows.

Definition 21 (t-$GSweep$ *coverage*) Suppose that a set of PoI $U = \{u_1, u_2, ..., u_n\}$ is given. Also, we have a set of static sensors $S = \{s_1, s_2, ..., s_p\}$ and a set of mobile sensors $M = \{m_1, m_2, ..., m_q\}$. Assume the speed of mobile sensors is equal. And we define the time interval of t is the *sweep period* of the set of points U. For a given time interval $t > 0$, the PoI u_i is t-$GSweep$ *covered* if an only if either of the following two cases is satisfied: (1) A static sensor is deployed at u_i which monitors u_i continuously. (2) At least one mobile sensor monitors u_i with every t time interval.

According to the definition of t-$GSweep$ coverage, we also define an energy efficient sweep coverage problem, *EEGSweep coverage* problem, as follows.

Definition 22 (*EEGSweep coverage*) Suppose that a set of PoI $U = \{u_1, u_2, ..., u_n\}$ is given. Also, we have a set of static sensors $S = \{s_1, s_2, ..., s_p\}$ and a set of mobile sensors $M = \{m_1, m_2, ..., m_q\}$. Assume that the speed of mobile sensors is same and λ, μ be the energy consumption at every time unit for a static sensor and mobile sensor. Then, the *EEGSweep coverage* problem is to find X static sensors ($X \leq p$) and Y mobile sensors ($Y \leq q$) such that every point PoI in U is t-$GSweep\ covered$ and $\lambda X + \mu Y$ is minimized.

Also, the following Theorem provides the complexity result of the *EEGSweep coverage* problem.

Theorem 9 *The EEGSweep coverage problem is NP-hard and it cannot be approximated with a factor of 2 unless* $P = NP$.

Proof When $\lambda = \mu$, the *EEGSweep coverage* problem is reducible to the *sweep coverage* problem by Li et al. [37]. The goal of *sweep coverage* problem is to minimize total number of sensors. In [37], authors showed the *sweep coverage* problem is NP-hard and cannot be approximated with a factor of 2 unless $P = NP$. Hence, we prove that the *EEGSweep coverage* problem is also NP-hard and cannot be approximated with a factor of 2 unless $P = NP$. □

3.3.2 *SEEGSweep Coverage* Problem

Moreover, Gorain and Mandal [40] introduced a special case of *EEGSweep coverage* problem which is called as *SEEGSweep coverage* problem that every mobile sensor visits the same subset of PoI to support t-$GSweep$ coverage while static sensors are applied to guarantee t-$GSweep$ coverage. Then, we formally define the *SEEGSweep coverage* problem as follows.

Definition 23 (*SEEGSweep coverage*) Assume that a set of PoI $U = \{u_1, u_2, ..., u_n\}$ is given. And, we have a set of static sensors $S = \{s_1, s_2, ..., s_p\}$ and a set of mobile sensors $M = \{m_1, m_2, ..., m_q\}$. Suppose that the moving speed of mobile sensors is equal and λ, μ be the energy consumption at every time unit for a static sensor and mobile sensor. Then, the *SEEGSweep coverage* problem is to find X static sensors and Y mobile sensors such that:

(1) X static sensors provide t-$GSweep$ coverage of X number of PoI ,

(2) The remaining PoI, $n - X$ are t-$GSweep$ covered by Y mobile sensors on condition that each mobile sensor monitors $n - X$ PoI,

(3) $\lambda X + \mu Y$ is minimized.

3.3.3 *ERSweep Coverage* Problem

Sensors have limited resource such as limited power battery, So, it is difficult to maintain *sweep coverage* for long-running applications without any recharge or any

replacement for batteries in sensors as well as to recharge or replace batteries often. Without loss of generality, after a specific time interval, those activities including replacement can be implemented. Until the time interval, the sensors should use their existing batteries to provide *sweep coverage*. Based on this observation, in [44], Gorain and Mandal also defined a *restricted sweep* ($ERSweep$) *coverage* problem.

3.3.4 *ERSweep Coverage* Problem

In *restricted sweep* ($ERSweep$) *coverage* problem, we consider there is an upper bound Z of energy utilization by a mobile sensor at every time interval τ. That is, a mobile sensor consumes maximum energy Z within every time interval $[\tau' + k\tau, \tau' + (k+1)\tau]$, where τ' is the starting time of movement of the mobile sensor and k are visiting points, $k \geq 0$. Also, it is noted that when a mobile sensor has the consumption of the energy Z before it completes a time interval, then the mobile sensor ceases its activities such as movement, visiting PoI until the forthcoming time interval.

Now, we formally define the $ERSweep$ *coverage* problem as follows.

Definition 24 (*ERSweep coverage*) Suppose that a set of PoI $U = \{u_1, u_2, ..., u_n\}$ is given. Also, we have a set of mobile sensors $M = \{m_1, m_2, ..., m_q\}$. Also, suppose the speed of mobile sensors is equal and let Z be the upper bound of energy consumption by a mobile sensor at every τ time interval. Let $\eta_i \geq 0$ be the energy consumption of a mobile sensor in order to monitor PoI u_i, where $i = 1, 2, ..., n$. Also, let μ be the energy consumption by a movement of mobile sensor per unit time. Then, the *ERSweep coverage* problem is to search optimal (or minimum) number of mobile sensors on condition that every PoI in U is t-$sweep$ covered when a given $t > 0$.

Furthermore, we note that when a mobile sensor has less than η_i energy before monitoring PoI u_i, the sensor partially finishes the visit and waits until the forthcoming time period. Also, Gorain and Mandal [44] provided the following Theorem.

Theorem 10 *The ERSweep coverage problem is NP-hard and it cannot be approximated with a factor of 2 unless* $P = NP$.

Proof If there is no visiting cost at PoI u_i, (i.e., $\eta_i = 0 for 1 \leq i \leq n$ and $t = \tau$, $\mu \cdot t = Z$, a mobile sensor's energy consumption within time t is Z that is the energy consumption because of the sensor's movement. In this particular case, the *ERSweep coverage* problem is to verify the minimum number of mobile sensors to provide a guaranteed *sweep coverage* with the set of PoI U, which it has been proved as NP-hard and cannot be approximated with a factor of 2 unless $P = NP$ by Li et al. [37]. Hence, we prove that the *ERSweep coverage* problem is also NP-hard and cannot be approximated with a factor of 2 unless $P = NP$. □

3.4 Sweep Coverage Problem with the Shorten Trajectory of Mobile Sensors

At previous research, the *sweep coverage* using a single mobile sensor is equivalent to the Euclidean Traveling Salesman Problem (TSP). But, the *sweep coverage* using multiple mobile sensor can be performed with the following ideas:

- For given PoI, we divide them into several PoI according to their locations. And, we compute distinct TSP cycles by each mobile sensor [38, 39, 45, 47].
- We compute a global TSP cycle according to the PoI and then divide it into several loop for each mobile sensor [48–50].

Feng et al. [51] found those two ideas may have disadvantages and neither of them do not provide the optimal solutions for *sweep coverage*. Figures 18 and 19 show the counterexamples. Suppose that a small circle is depicted as PoI and a small triangle represents a mobile sensor. Figure 18a shows two mobile sensors cover one TSP cycle together with a total trajectory length of 6. But, as Fig. 18b, if the mobile sensors have two separate cycles, then it is possible the trajectory length is reduced to 4. On the other hand, Fig. 19a describes that 6 PoI are sweep covered by two mobile sensors with 2 distinct cycles. However, as Fig. 19b, we may have a better way with a collaborate work in only one TSP cycle. From the examples, it is observed that either working together with one TSP cycle or working separately with distinct cycles may not give the optimal solution.

Feng et al. [51] considers the sweep coverage in which mobile sensors are able to move along the equal trajectory or isolated trajectory. If all mobile sensors work together, then the longest path decides the scanning interval referred as *makespan*. Also, if we can reduce the length of the *makespan* path, then we also can shorten the scanning interval. Based on this observation, Feng et al. [51] introduce M^3SR problem whose objective is to minimize the makespan of the mobile trajectories.

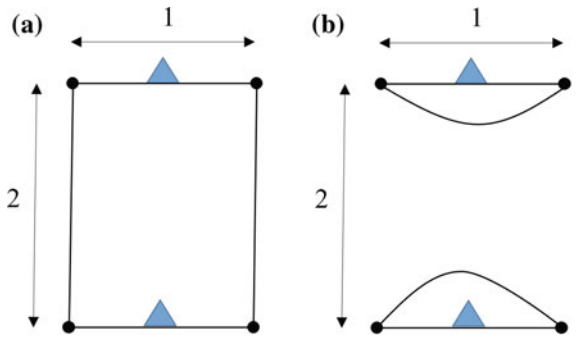


Fig. 18 A counterexample 1

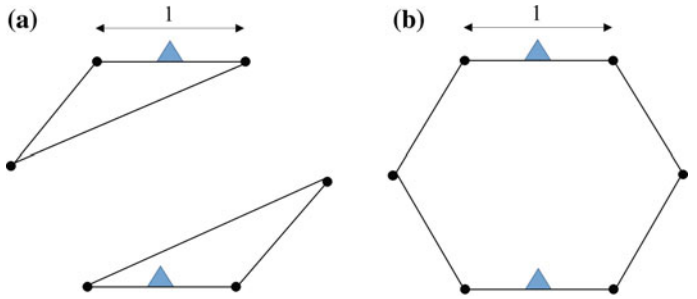


Fig. 19 A counterexample 2

3.4.1 *ERSweep Coverage* Problem

Suppose that we have a set of PoI $P = \{p_1, p_2, ..., p_n\}$ and a set of mobile sensors $S = \{s_1, s_2, ..., s_m\}$ in the given field. Also, suppose the position of each PoI is given. Then, we can define that PoI p_i is covered if an only a mobile sensor s_j takes to p_i. Assume the speed of mobile sensor is equal and mobile sensors can move along the equal or different trajectories with other sensors. Assume also if several sensors move along the equal path, then the length of sweep route will be divided equally. Note that our objective is to minimize the sweep coverage duration that is equivalent to minimize the longest sweep trajectory. Now, we define M^3SR problem as follows.

Definition 25 (M^3SR) Given the positions of n PoI and m mobile sensors, the *Minimum Makespan of Mobile Sweep Routes* problem (M^3SR problem) is to decide the schedule of mobile sweep routes in order to shorten the makespan.

Also, for $q \in \{1, 2, ..., m\}$, we formulate the M^3SR problem as follows.

$$\textbf{Minimize}\ weight_q(P) = max_{i=1...q}\left(\frac{TSP(P_q^i)}{|S_q^i|}\right) \tag{25}$$

Subject to:

$$\bigsqcup_{i=1}^{q} S_q^i = S, (S_q^i \neq 0) \tag{26}$$

$$S_q^i \bigcap_{i \neq j} S_q^j = 0, (i, j \in \{1, 2, ..., q\}) \tag{27}$$

$$\bigsqcup_{i=1}^{q} P_q^i = P, (P_q^i \neq 0) \tag{28}$$

$$P_q^i \bigcap_{i \neq j} P_q^j = 0, (i, j \in \{1, 2, ..., q\}) \tag{29}$$

We partition every PoI and mobile sensors into q classes ($q \in \{1, 2, ..., m\}$). P_q^i and S_q^i represent the set of PoI and the set of mobile sensors which is related to ith class. Suppose $TSP(P_q^i)$ is the lenth of the TSP cycle for PoI within ith class and $weight_q(P)$ means the makespan of mobile sweep trajectories if every PoI and mobile sensor is related to q distinct routes. Also, for each q, we get one $weight_q(P)$ and the minimum of those $weight_q(P)$ should be a solution of M^3SR.

In addition, we show the 2D version of M^3SR is NP-hard as follows.

Theorem 11 *The 2D version of M^3SR is NP-hard.*

Proof Let us consider a special case, $m = 1$. Then, M^3SR in 2D case is equivalent to search for a shortest sweep route for a single mobile sensor to monitor every PoI. It is a well known TSP which is NP-hard. It follows that the TSP is a special case of the M^3SR in 2D case. Hence, we proved that M^3SR is NP-hard. □

It has been known that it is difficult to find a TSP within polynomial time. However, if think the relationship between Traveling Salesman Problem and Minimum Spanning Tree Problem, it is possible to replace the $TSP(P_q^i)$ with $MST(P_q^i)$. Then, the MST can be converted into TSP cycle. Based on this, we also define a new problem referred as MST-M^3SR as follows.

Definition 26 (MST-M^3SR) MST-M^3SR is different from MST-M^3SR problem. The difference is that $weight(P) = max_{i=1...q}(\frac{MST(P_q^i)}{|S_q^i|})$ in MST-M^3SR problem, where $MST(P_q^i)$ is the weight of the minimum spanning tree for PoI in P_q^i.

3.5 Target Coverage and Network Connectivity Using Mobile Sensors

A coverage of interest point and network connectivity can be considered as main challenging issues in wireless sensor networks. Using the mobility of sensors, Liao et al. [52] introduced the mobile sensor deployment problem, called as MSD problem, which requires careful deployment of mobile sensors in order to provide *target coverage* and *network connectivity* with a minimum movement.

3.5.1 System Model

Let us assume we have the set of targets $T = \{t_1, t_2, ..., t_m\}$ which are known positions to be covered. Also, we have the set of mobile sensors $S = \{s_1, s_2, ..., s_n\}$ which are deployed randomly in the given area. Then, the system model is defined as follows.

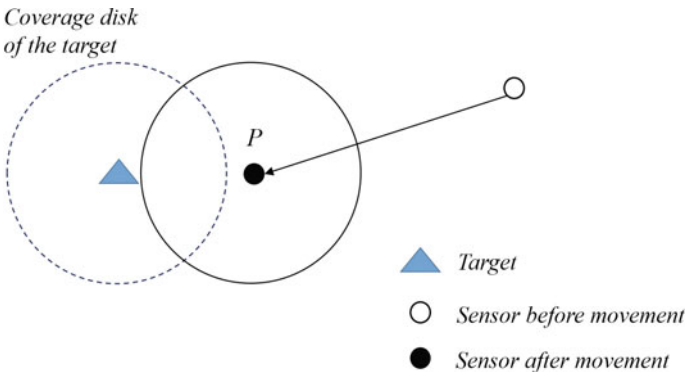


Fig. 20 A example of the straight-line movement of mobile sensor from its initial location and the target (destination P) for minimizing the moving distance

- Using GPS or a localization algorithm, each mobile sensor can estimate own location. Also, the control center such as sink collects the information of sensors and broadcasts the requests of movements to mobile sensors.
- We basically consider the obstacle-free area for the movement. If there exists obstacles, we assume a sensor can decide an proper shortest trajectory to the destination so as to bypass the obstacles on the path.
- By Bai et al. [53], the disk model is applied for two types of ranges: a sensing range r_s and a communication range r_c. A target is said to be covered if and only if there is at least one sensor within the sensing range r_s centered at the target.
- By Tan et al. [54], the free mobility model is used in the system. So, sensors are basically able to move to any direction and to stop at any location. A movement distance of sensor is related to the energy consumption of sensor. So, more movement results in more energy consumption of sensor. The movement distance of sensor s to monitor target t is defined as $dist(s,t) - r_s$, where $dist(s,t)$ is the Euclidean distance between s and t. Also, the $dist(s_i, s_j) - r_c$ is considered as the movement distance for a connection between sensor s_i and s_j. With the assumption of the obstacle-free area for movement, a sensor has a straight-line movement from initial location to the target position until it touches the target's coverage circle. Figure 20 shows an example of the straight-line movement of mobile sensor from initial position to the target position in order to minimize the movement distance.

3.5.2 *MSD* Problem

Based on the system model, Liao et al. [52] defined MSD problem. It is given as follows.

Definition 27 (MSD) Suppose we have m number of targets and n number of mobile sensors have initial random locations in the task area. The MSD problem is to

minimize the movement of mobile sensors on condition that the following objectives are performed after mobile sensors visit new locations:

(1) Each target is to be covered by at least one mobile sensor.
(2) By movements of all the moved sensors, the network is connected.

As defined in Definition 27, the MSD problem deals with two issues: *target coverage* and *network connectivity*. It follows that the MSD problem can be partitioned into two sub-problems. So, it is possible we conquer them one by one. To solve the problems, we first consider how to deploy mobile sensors to cover targets with minimal movement distance. We define those sensors are *coverage sensors*. Then, we consider the deployment of the rest sensors to support connectivity between coverage sensors and the control center such as a sink.

Then, we formally define two sub-problems as follows.

Definition 28 ($TCOV$) Assume we have m number of targets and n number of mobile sensors have initial random locations in the task area. the $TCOV$ problem is to move sensors to new locations on condition that every target is to be covered and the total moving distance of mobile sensors is minimized.

Definition 29 ($NCON$) Suppose that we have a sink, the set of coverage sensors and the rest mobile sensors after the $TCOV$ problem is conquered. The $NCON$ problem is to find the deployment of the rest mobile sensors in order to provide the connectivity between coverage sensors and the sink with a minimum moving distance.

Now, we show $TCOV$ problem is NP-hard. To do it, we first consider a special case of $TCOV$, referred as $TCOV'$ and prove $TCOV'$ is NP-hard. It follows that we can induce the NP-hardness of the original $TCOV$ problem. The $TCOV'$ problem can be considered as a special case of $TCOV$ that all the sensors have same positions initially. That is, mobile sensor has same starting positions. Then, if there exists a solution of $TCOV$, the $TCOV'$ problem can be solved by locating mobile sensors with same initial positions, but the converse cannot be followed.

Theorem 12 *The TCOV′ problem is NP-hard.*

Proof Let the power set of T be $P(T)$. Note that totally, there are $n * 2^m$ number of potential positions for the n mobile sensors in $TCOV'$ problem. Then, it is said that each potential position corresponds to a subset of T. For each potential position, a weight W can be assigned as its corresponding element in $P(T)$, which can be considered as the moving distance between the mobile sensor and the corresponding potential position.

Then, the decision version of the $TCOV'$ can be converted into the below *set cover* problem.

Given the set of targets T and a finite number of weighted subsets of T whose union consists of the universe, the *set cover* problem is to decide if there exist some subsets whose total weight is less than equal to W on condition that the union of those subsets includes all elements within T.

It follows that the decision version of $TCOV'$ is equivalent to the weighted set cover problem [55] that has been known as NP-complete. Hence, we prove that $TCOV'$ is NP-hard. □

By the above Theorem 12 and its proof, it is proved that $TCOV$ is also NP-hard since $TCOV'$ is a special case of $TCOV$.

References

1. Huang, C.F., Tseng, Y.C.: The coverage problem in a wireless sensor network. In: Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications (WSNA) (2003)
2. Kumar, S., Lai, T.H., Balogh, J.: On k-coverage in a mostly sleeping sensor network. In: Proceedings of ACM Annual International Conference on Mobile Computing and Networking (MobiCom) (2004)
3. Cardei, M., Wu, J.: Energy-efficient coverage problems in wireless ad hoc sensor networks. J. Comput. Commun. Sens. Netw. (2005)
4. Meguerdichian, S., Koushanfar, F., Potkonjak, M., Srivastava, M.B.: Coverage problems in wireless ad-hoc sensor networks. In: Proceedings of IEEE Conference on Computer Communications (INFOCOM) (2001)
5. Lin, L., Lee, H.: Distributed algorithms for dynamic coverage in sensor networks. Proceedings of ACM Symposium on Principles of Distributed Computing (2007)
6. Gage, D.: Command control for many-robot systems. In: Proceedings of the Nineteenth Annual AUVS Technical Symposium (AUVS-92) (1992)
7. Howard, A., Mataric, M.J., Sukhatme, G.S.: Mobile sensor network deployment using potential fields: a distributed, scalable solution to the area. In: Proceedings of the 6th International Symposium on Distributed Autonomous Robotic Systems (DARS), pp. 299–308. Fukuoka, Japan (2002)
8. Liu, B., Brass, P., Dousse, O., Nain, P., Towsley, D.: Mobility improves coverage of sensor networks. In: Proceedings of the 6th IACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Illinois, USA (2005)
9. Hall, P.: Introduction to the Theory of Coverage Processes. Wiley (1988)
10. Poduri, S., Sukhatme, G.S.: Constrainded coverage for mobile sensor networks. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 165–172. New Orleans, USA (2004)
11. Lambrou, T.P.: Optimized cooperative dynamic coverage in mixed sensor networks. ACM Trans. Sens. Netw. 1–46 (2015)
12. Papadimitriou, C.: The euclidean travelling salesman problem is NP-complete. Theor. Comput. Sci. 237–244 (1977)
13. Kumar, S., Lai, T.H., Arora, A.: Barrier coverage with wireless sensors. In: Proceedings of the 11th Annual International Conference on Mobile Computing and Networking (MobiCom), pp. 284–298 (2005)
14. Saipulla, A., Westphal, C., Liu, B., Wang, J.: Barrier coverage of line-based deployed wireless sensor networks. In: Proceedings of the 28th IEEE Conference on Computer Communications (INFOCOM) (2009)
15. Kumar, S., Lai, T.H., Posner, M.E., Sinha, P.: Maximizing the lifetime of a barrier of wireless sensors. IEEE Trans. Mob. Comput. (TMC) **9**(8) (2010)
16. Liu, B., Dousse, O., Wang, J., Saipulla, A.: Strong barrier coverage of wireless sensor networks. In: Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc) (2008)

17. Li, L., Zhang, B., Shen, X., Zheng, J., Yao, Z.: A study on the weak barrier coverage problem in wireless sensor networks. Comput. Netw. **55**(3), 711–721 (2011)
18. Chen, J., Li, J., Lai, T.H.: Energy-efficient intrusion detection with a barrier of probabilistic sensors: global and local. IEEE Trans. Wirel. Commun. **12**(9), 4742–4755 (2013)
19. Chen, A., Zhu, Y., Li, Z., Lai, T.H., Liu, C.: Is one-way barrier coverage achievable using comprehensive sensors? Comput. Commun. **57**, 100–114 (2015)
20. Ban, D., Jiang, J., Yang, W., Dou, W., Yi, H.: Strong k-barrier coverage with mobile sensors. In: Proceedings of the IEEE International Wireless Communications and Mobile Computing Conference (IWCMC), France (2010)
21. Lawler, E.: Combinatorial optimization: networks and matroids. Holt, Renehard and Winston Press, New York (1976)
22. Chen, D.Z., Gu, Y., Li, J., Wang, H.: Algorithms on minimizing the maximum sensor movement for barrier coverage of a linear domain. Discret. Comput. Geom. **50**(2), 374–408 (2013)
23. Li, S., Shen, H.: Minimizing the maximum sensor movement for barrier coverage in the plane. In: Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), Hong Cong (2015)
24. Burkard, R.E., Cela, E.: Linear Assignment Problems and Extensions. Springer (1999)
25. Silvestri, S., Goss, K.: Mobibar: an autonomous deployment algorithm for barrier coverage with mobile sensors. Ad Hoc Netw. **54**, 111–129 (2017)
26. Kim, H., Son, J., Chang, H.J., Oh, H.: Event-driven partial barriers in wireless sensor networks. In: Proceedings of the IEEE International Conference on Computing, Networking and Communications (ICNC), Hawaii, USA, Feb 2016
27. Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems. J. ACM **19**(2), 248–264 (1972)
28. Kim, H., Ben-Othman, J.: On resilient event-driven partial barriers in mobile sensor networks. In: Proceedings of the IEEE International Conference on Communications (ICC), Malaysia, May 2016
29. Wang, G., Cao, G., Porta, T.L., Zhang, W.: Sensor relocation in mobile sensor networks. In: Proceedings of IEEE International Conference on Computer Communications (INFOCOM), Miami, FL, March 2005
30. Akkaya, K., Senel, F.: Detecting and connecting disjoint sub-networks in wireless sensor and actor networks. Ad Hoc Netw. J. (Elsevier) **7**(7), 1330–1346 (2009)
31. Akkaya, K., Guneydas, I., Bicak, A.: Autonomous actor positioning in wireless sensor and actor networks using stable-matching. Int. J. Parallel Emerg. Distrib. Syst. (IJPEDS) **25**(6), 439–464 (2010)
32. Abbasi, A., Younis, M., Akkaya, K.: Movement assisted connectivity restoration in wireless sensor and actor networks. IEEE Trans. Parallel Distrib. Syst. **20**(9), 1366–1379 (2009)
33. Wang, S., Mao, X., Tang, S., Li, X., Zhao, J., Dai, G.: On movement-assisted connectivity restoration in wireless sensor and actor networks. IEEE Trans. Parallel Distrib. Syst. **22**(4), 687–694 (2011)
34. Kloder, S., Hutchinson, S.: Barrier coverage for variable bounded-range line-of-sight guards. In: Proceedings of the IEEE International Conference on Robotics and Automaton, Roma, Italy, April 2007
35. Bhattacharya, B., Burmester, M., Hu, Y., Kranakis, E., Shi, Q., Wiese, A.: Optimal movement of mobile sensors for barrier coverage of a planar region. Theor. Comput. Sci. (Elsevier), pp. 5515–5528 (2009)
36. Cheng, W., Li, M., Liu, K., He, Y., Li, X., Liao, X.: Sweep coverage with mobile sensors. In: Proceedings of the IEEE International Symposium on Parallel and Distributed Processing (IPDPS), pp. 1–9. Miami, Florida, USA (2008)
37. Li, M., Cheng, W., Liu, K., He, Y., Li, X., Liao, X.: Sweep coverage with mobile sensors. IEEE Trans. Mob. Comput. (TMC) **10**(11), 1534–1545 (2011)
38. Du, J., Li, Y., Liu, H., Sha, K.: On sweep coverage with minimum mobile sensors. In: Proceedings of IEEE International Conference on Parallel and Distributed Systems (ICPADS), pp. 283–290, Shanghai, China (2010)

39. Liu, B.H., Nguyen, N.T., Pham, V.T.: An efficient method for sweep coverage with minimum mobile sensor. In: Proceedings of IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), pp. 289–292. Kitakyushu, Japan (2014)
40. Gorain, B., Mandal, P.S.: Approximation algorithms for sweep coverage in wireless sensor networks. J. Parallel Distrib. Comput. (Elsevier) **74**, 2699–2707 (2014)
41. Li, J., Wang, R., Huang, H., Sun, L.: Voronoi-based coverage optimiation for directional sensor networks. Wirel. Sens. Netw. 417–424 (2009)
42. Gorain, B., Mandal, P.S.: Brief acnnouncement: sweep coverage with mobile and static sensors. In: Proceedings of International Symposium on Stabilization, Safety, and Security of Distriuted Systems (SSS 2014), pp. 346–348. Paderborn, Germany (2014)
43. Gorain, B., Mandal,P.S.: Energy efficient sweep coverage with mobile and static sensors. In: Proceedings of Conference on Algorithms and Discrete Applied Mathematics (CALDAM), pp. 275–285. LNCS (2015)
44. Gorain, B., Mandal, P.S.: Solving energy issues for sweep coverage in wireless sensor networks. Discret. Appl. Math. (Elsevier) (2016)
45. Shu, L., Cheng, K., Zhang, X., Zhou, J.: Periodic sweep coverage scheme based on periodic vehicle routing problem. J. Netw. **9**(3), 726–732 (2014)
46. Tiwari, S.O., Kumar Yadav, S.: Data harvesting with mobile elements in wireless sensor networks. J. Electron. Commun. Eng. **9**(1), 104–114 (2014)
47. Gu, Y., Bozdag, D., Brewer, R.W., Ekici, E.: Data harvesting with mobile elements in wireless sensor networks. Comput. Netw. **50**(17), 3449–3465 (2006)
48. Zhao, D., Ma, H., Liu, L.: Mobile sensor scheduling for timely sweep coverage. In: Proceedings of IEEE Wireless Communications and Networking Conference (WCNC), pp. 1771–1776. Shanghai, China (2012)
49. Moazzez-Estanjini, R., Paschalidis, I.C.: Improved delay-minimized data harvesting with mobile elements in wireless sensor networks. In: Proceedings of IEEE International Symposium oon Modeling, Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), pp. 49–54. Princeton, USA (2011)
50. Moazzez-Estanjini, R., Paschalidis, I.C.: On delay-minimized data harvesting with mobile elements in wireless sensor networks. Ad Hoc Netw. **10**(7), 1191–1203 (2012)
51. Feng, Y., Gao, X., Wu, F., Chen, G.: Shorten the trajectory of mobile sensors in sweep coverage problem. In: Proceedings of IEEE GLOBECOM (2015)
52. Liao, Z., Wang, J., Zhang, S., Cao, J., Min, G.: Minimizing movement for target coverage and network connectivity in mobile sensor networks. IEEE Trans. Parallel Distrib. Syst. **26**(7), 1971–1983 (2015)
53. Bai, X., Kumar, S., Xuan, D., Yun, Z., Lai, T.H.: Deploying wireless sensors to achieve both coverage and connectivity. In: Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc), pp. 131–142 (2006)
54. Tan, R., Xing, G., Wang, J., So, H.C.: Exploiting reactive mobility for collaborative target detection in wireles sensor networks. IEEE Trans. Mob. Comput. pp. 317–332 (2010)
55. Karp, R.M.: Reducibility Among Combinatorial Problems. Springer (1972)

Part III
Task Allocation and Mission Assignment

Energy-Aware Task Allocation in WSNs

Wanli Yu, Yanqiu Huang and Alberto Garcia-Ortiz

Abstract Complex wireless sensor network applications as those in Internet of Things or in-network processing are pushing the requirements for energy efficiency and data processing drastically. Executing the tasks of such complex applications in a single node may lead it to die soon, since the nodes in WSNs are usually with limited and generally irreplaceable power sources. How to distribute the tasks across the network and simultaneously balance the energy consumption of each node to achieve energy efficiency and to extend the network lifetime are crucial and urgent requirements in WSNs. Energy-aware task allocation (sometimes also called workload distribution) technologies, which have been deeply studied in multiprocessor systems, grid computing, and system on chip (SoC), are attracting the attention of the research community in WSNs. Due to the limited energy source and computing capability as well as the wireless communication, the task allocation problem in WSNs is different from traditional wired systems. This chapter provides an application-level taxonomy and an in-depth review of task allocation approaches in WSNs. It enables the readers to gain a clear view of current task allocation approaches, by taking the evaluation metrics and the modeling methods of the problem into account.

W. Yu · Y. Huang · A. Garcia-Ortiz (✉)
University of Bremen, Bremen, Germany
e-mail: agarcia@item.uni-bremen.de

W. Yu
e-mail: wyu@item.uni-bremen.de

Y. Huang
e-mail: huang@item.uni-bremen.de

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_6

1 Introduction

Wireless sensor networks (WSNs) are composed of a larger number of low-cost and low-power sensor nodes which are spatially distributed to monitor the physical or environmental conditions. This key technology nowadays has been applied to a wide variety of applications, such as monitoring of physical environments, enhanced industrial control, remote health care, logistic, with vastly varying requirements and characteristics. In almost any WSN application, energy efficiency is considered to be a primary concern: The sensor nodes in WSNs are usually supplied by the limited battery power; it is hard to recharge or replace the dying sensor nodes due to large quantities or the harsh physical environments, which would lead to fragmentations of the network and loss of potentially critical information. Therefore, many research and industrial communities are devoted to studying how to achieve energy efficiency and to extend the network lifetime of WSNs.

In traditional WSN applications, the workloads are very simple and wireless communication is usually the most energy-intensive process [1]. Specifically, a single-bit transmission requires 1000 times the energy cost of a 32-bit computation in a classical architecture [2]. Thus, most of the previous researches mainly focus on reducing the communication cost at the expense of increasing the computation cost. For example, energy-efficient clustering and routing approaches have been proposed to conserve communication energy by reducing the transmission distance and balancing the transmission loads within the clusters [3–5]. Alternatively, there are numerous compression-based techniques that either focus on reducing the volumes of the transmitting packets [6–8] or aim to decrease the transmission rate to achieve the energy efficiency [9–11]. Also, a large number of sleep/wake-up schemes have been studied to reduce the energy spent on idle states of the radio component [12, 13].

However, as more complex applications have been implemented in WSNs during the past decade, such as Internet of Things or in-network processing, the computation energy consumption is comparable with or even larger than the communication cost [14, 15]. Executing those computationally intensive applications may make the sensor nodes die soon if the workloads of the tasks are not fairly distributed. In order to achieve energy efficiency and to extend the network lifetime, it is necessary to consider and balance properly the workload of not only the communication tasks but also the sensing and processing tasks among the sensor nodes. Energy-aware task allocation, as one efficient solution to solve the energy balance problems, starts to attract the attention of WSN research community. Although task allocation approaches have been deeply studied in multiprocessor systems, grid computing, and system on chip (SoC), the limited battery power and computing capability of sensor nodes as well as the wireless communication in WSNs make the problems difficult.

This chapter presents an application-level taxonomy and an in-depth analysis of the task allocation approaches in WSNs. The structure is organized as follows: It firstly studies the important evaluation metrics of the task allocation problem in Sect. 2; then Sect. 3 presents the models of the tasks, networks, and cost functions of the WSN nodes; the discussion of the classification of the task allocation approaches

is shown in Sect. 4; the related detailed information of the approaches is presented in Sects. 5 and 6, respectively; the conclusion is given in the last section.

1.1 Main Objectives and Challenges of Task Allocation in WSNs

As mentioned earlier, the sensor nodes in WSNs are usually powered by limited battery sources which are hard to recharge or replace. The straightforward objective is to prolong the network lifetime. It does not mean just focusing on reducing the energy consumption of a single node, the energy cost of all of the sensor nodes in the network should be considered and balanced. If an improper task allocation scheme is used, some sensor nodes in the network would quickly run out of energy because of the overload, even if other sensor nodes still have plenty of residual energy. As a result, the network cannot provide enough information for adequate time duration. Thus, a good task allocation algorithm should be able to extend the network lifetime by properly assigning the tasks of the application among the sensor nodes.

In addition to extend the network lifetime, a good task allocation algorithm also needs to guarantee that the application can be completed before the deadlines. Especially, when some emergent scenarios appear, e.g., fire hazard or military information, even very short time delay could lead to an accident. Besides the overall time requirement, it is also necessary to consider the individual time requirement of each sensor node when executing the assigned tasks to better match the media access control (MAC) and routing protocols. For example, when the time-division multiple access (TDMA) is applied, each sensor node has to finish its tasks before transmitting at the settled time slot.

As seen, task allocation in WSNs is a complex process with multiple heterogeneous requirements. It needs to conserve the limited resources of the sensor nodes and to be very energy efficient. Currently, most of the task allocation algorithms only provide static task allocation solutions, which is not sufficient for the dynamic networks. For example, when some new sensor nodes are added to the network, a fast feedback is required to reassign the tasks to the sensor nodes. Therefore, how to design an online task allocation algorithm with less complexity to adapt the network changes is a main challenge.

Although the traditional centralized algorithms are easy to implement, a centralized decision maker has to collect all of the information of the network, which could result in a large time delay. Furthermore, the robustness of the algorithms has to be considered because of the instability of the wireless links and the accidental failure or block of the sensor nodes. To this end, distributed task allocation algorithm which is able to provide fault-tolerant mechanism is one of the most important trends in the future.

2 The Evaluation Metrics of Task Allocation Approaches

This section introduces the metrics which are used to evaluate the performance of the task allocation approaches. The metrics utilized by most research works mainly include network lifetime, network energy consumption and residual energy distribution, time requirement and fault tolerance.

- **Network Lifetime**
  Network lifetime is one of the most important characteristics in WSNs. Due to the energy constraints as analyzed in Sect. 1, the straightforward evaluation metric is how long the network lifetime can be extended by using the task allocation algorithms. There are many parameters influencing network lifetime, e.g., the number of the alive sensor nodes in the network, the coverage and connectivity of the network. Therefore, the precise definitions of the network lifetime (NL) vary in relevant literature. The common definitions are presented as follows:

(a) Defining NL according to the number of died sensor nodes: NL is defined as the time when k ($1 \leq k \leq K$) out the total K sensor nodes in the network die.
(b) Defining NL according to the network coverage: When the region of interest is covered by less than k sensor nodes, the network ends.
(c) Defining NL according to the network connectivity: The network is considered to die when there are less than k sensor nodes which have a path to the sink node.
(d) Defining NL according to the quality of service: NL is defined as the time until the network cannot guarantee the application requirements.

  Although there are so many different definitions of the network lifetime, the most frequently used definition in existing literature is k out K sensor nodes die. Especially when $k = 1$, which means the network lifetime is defined as the time when the first sensor node runs out of energy. In this chapter, we mainly focus on the studies which are using this network lifetime definition.
- **Network Energy Consumption and Residual Energy Distribution**
  In addition to the metric of network lifetime, the overall network energy consumption and residual energy distribution are also preferred by many researches to estimate the performances of the task allocation approaches. Several studies assume that a certain amount of energy can be used for the whole network. The network is out of functionality when this energy is completely depleted [16, 17]. Therefore, minimizing the summation of all sensor nodes' energy consumption (termed as network energy consumption) is the main focus. In addition, the standard deviation of the residual energy of each sensor node is used by a number of publications to evaluate the task allocation methods [18–21]. A smaller value of the residual energy distribution represents more balanced lifetime of each sensor node, which indicates a better performance of the methods.
- **Time Requirement**
  In WSNs, the time requirement is another important metric that needs to be considered [16, 20–24]. In many WSN applications, it is mandatory to know the

presence of some events fast to make a quick response. The time requirements of different applications vary. Typically, the makespan from the first task being executed to the last one being completed should be no longer than a prescribed time. Moreover, the time requirement should also be satisfied for specific sensor network communication protocols.

- **Fault Tolerance**
  The reliability of the performance of sensor nodes in WSNs is mainly subject to several crucial impact factors, such as the wireless link errors, the hardware (sensing unit or wireless transceiver) failures and malicious attacks. In order to better understand the monitored physical or environmental scenarios, fault tolerance is one of the critical issues that needs to be considered in task allocation problem for WSNs [17].

3 Modeling Application Tasks, Networks and Cost Functions of Sensor Nodes

The models of the application tasks, networks and cost functions of WSN nodes are the bases for formulating the task allocation problem. The accuracy of the energy model especially affects the performance of the approaches when they are used in reality. This section presents these basic modelings, and the related notations used for the modelings are listed in Table 1.

3.1 Application Tasks Model

Typically, a WSN application is made up of a set of computationally in-network processing tasks [15, 16, 23–25]. The corresponding tasks and their relationships

Table 1 Notations used for the modelings

Notation	The meaning of each notation
v_i	The ith vertex (task) of the modeled DAG graph (application)
w_i	The computational workload of task v_i
e_{ij}	The communication from task v_i to v_j
l_{ij}	The amount of transmitted data from task v_i to v_j
N_k	The kth node
f_k, P_k	The processing speed and power of node N_k
t_{p_ki},e_{p_ki}	The execution time and energy cost of node N_k for executing task v_i
t_{o_k}, e_{o_k}	The time duration and energy cost of node N_k for communication overhead activities
t_{cmm_k}	The communication time of node N_k
E_{tx_k}, E_{rx_k}	The transmitting and receiving energy cost of node N_k
t_{rel_k}, E_{rel_k}	The relay time duration and energy cost of node N_k

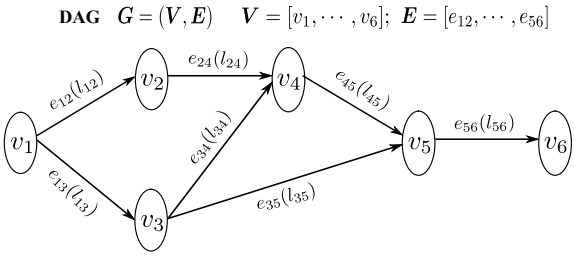


Fig. 1 An example of a directed acyclic graph (DAG)

can be modeled as a directed acyclic graph (DAG). By analyzing the well-designed data flow of the specific WSN application, the energy efficiency can be optimized accordingly.

Figure 1 shows one example of a DAG graph, $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V} = [v_1, \ldots, v_6]$ and $\boldsymbol{E} = [e_{12}, \ldots, e_{56}]$. Each vertex $v_i \in \boldsymbol{V}$ represents a task which is connected to others by directed edges. The computational workload of v_i is the total number of CPU clock cycles needed to execute the task, which is denoted as w_i. Each directed edge $e_{ij} \in \boldsymbol{E}$ stands for the communication from task v_i to v_j. The weight on e_{ij}, l_{ij}, usually represents the amount of transmitted data in bits.

Each task v_i consumes the data received from its predecessors, generates the data and transmits to its successors. Given a directed edge e_{ij}, v_i is called the predecessor of v_j, and v_j is the successor of v_i. In this case, task v_i has the higher priority than v_j, which means that the successor cannot be executed until it receives the data from all of its predecessors. In Fig. 1, for example, if task v_5 needs to be executed, it has to firstly receive the data from its predecessors v_3 and v_4. A task without any predecessor is called *source task*, which is typically assumed as the sensing task.[1] It does not have any predecessor and has the highest priority. Note that the requirements of the sensing assignments vary for different applications. It also should be noted that only the sensor nodes execute the sensing tasks.

3.2 Network Model

A WSN is composed of a set of sensor nodes (e.g., K sensor nodes) deployed in the monitoring area. For generality, we consider these K sensor nodes are heterogeneous, i.e., they could have different processing speeds, battery capacities and transmission ranges. The sensor nodes are connected by either single hop or multi-hop wireless communications according to different network structures. For concreteness, we introduce two typical types of network structures: multi-hop mesh and hierarchical cluster networks.[2]

[1] The number of the *source tasks* are not limited to one, it is determined by specific applications.

[2] Note that, the network structures of WSNs are not limited to multi-hop mesh and hierarchical cluster, there are also other types, such as location-based network structures.

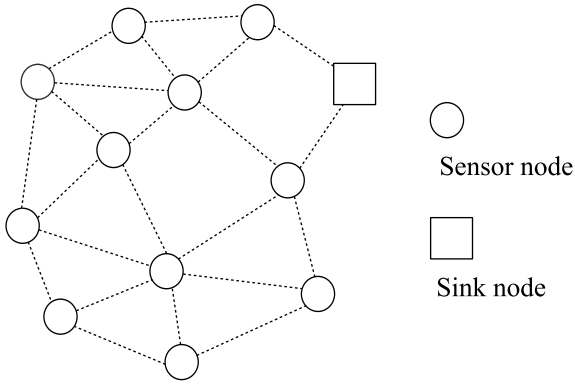


Fig. 2 A multi-hop mesh WSN (the dotted lines represent the wireless hops)

- Multi-hop Mesh WSN
  In multi-hop mesh WSNs, all the sensor nodes are considered equal with respect to their roles and functionalities. In addition to the abilities of sensing, processing, and transmitting, each sensor node also can operate as a relay node. The observation of each sensor node is propagated by multiple wireless hops from the first relay node to the next one until the observation reaches the destination (sink node). All of the neighbor nodes within the transmission range of the sender are able to act as the relay node. The selection of the transmission path depends on the used routing protocol. A multi-hop mesh WSN is shown in Fig. 2.
- Hierarchical Cluster WSN
  In hierarchical cluster WSNs, all the sensor nodes are grouped into different clusters and each cluster is usually made up of one leader node (cluster head) and several leaf nodes as shown in Fig. 3. The workload of the application is completed through the cooperation of the leaf nodes and the cluster head. Each leaf node is in charge of sensing and then either directly transmitting the raw data to the cluster head or preprocessing the data before the transmission. The cluster head is in charge of receiving data and executing further processing; after that, it forwards the processed data to the sink node by either direct transmission or multiple hops among the cluster heads. Typically, the sink node does not have energy constraints and each cluster works independently. Many studies mainly focus on the activities within the cluster [8, 15, 23].

3.3 Cost Functions

The network lifetime depends on the energy consumption of each node in the network. We formulate the energy cost of the activities of the nodes in this section.

In multi-hop mesh WSN, each node performs the execution of the tasks (either sensing or processing or both of them) and transmitting, also they can work as the relay nodes to receive the neighbors' information and forward them. Besides, when the node completes all of the assigned tasks, it goes to sleep mode to save energy.

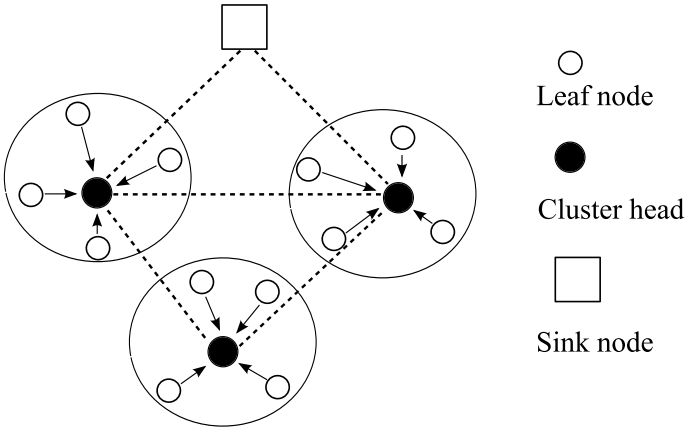


Fig. 3 A hierarchical cluster-based WSN (dotted lines represent the wireless hops among the cluster heads)

For generality, we consider that each node may have different energy parameters. Assuming there are K nodes $(N_1, \ldots, N_k, \ldots, N_K)$ in the network, the execution time of node N_k for executing the task v_i is:

$$t_{p\_ki} = \frac{w_i}{f_k} \tag{1}$$

where w_i is the processing workload of task v_i and f_k (Hz) is the processing speed of node N_k. Correspondingly, the energy consumption can be formulated as:

$$e_{p\_ki} = P_k \, t_{p\_ki} \tag{2}$$

where P_k (unit: J/s) is the average power consumption of node N_k.

As reported in [23], the communication cost of a WSN node includes the energy cost of not only the data packets communication and but also the overhead activities. The activities of the communication overhead consist of radio start-up, channel accessing, control packets, turnaround, idle listening, overhearing, and collision, which can sometimes have a major impact on the communication consumption. Thus, the one-hop communication time of node N_k is:

$$t_{cmm\_k} = t_{o\_k} + \frac{L}{Bandwidth} \tag{3}$$

where t_{o_k} is the time duration of the overhead activities and L is the amount of the data to be transmitted. According to the most popularly used communication energy model [23, 26], the cost of node N_k for transmitting and receiving L bits of data, E_{tx_k} and E_{rx_k}, can be expressed as:

$$\begin{aligned} E_{tx\_k} &= e_{o\_k} + (e_{elc} + \varepsilon_{amp}\, d^{\alpha})\, L = e_{o\_k} + e_{tx}\, L \\ E_{rx\_k} &= e_{o\_k} + e_{elc}\, L = e_{o\_k} + e_{rx}\, L \end{aligned} \tag{4}$$

where e_{o_k} is the energy consumption of the overhead activities; e_{elc} is the energy dissipated by the electronic circuits of the transceiver to transmit or receive 1 bit of data; ε_{amp} is the energy cost of the transmitter amplifier for transmitting 1 bit of data at 1 m distance; and α is the path loss exponent which is typically in the range $2 \leq \alpha \leq 4$ decided by the different environments [27]. Note that, if there are multiple tasks executed in node N_k, the inter-node communications among these tasks can be avoided.

When node N_k operates as a relay node, it firstly receives the data from the previous sensor node and then transmits the data to next one. The time and energy consumption of N_k when it executes the relay workloads, t_{rel_k} and E_{rel_k}, can be expressed as:

$$\begin{aligned} t_{rel\_k} &= 2\, t_{cmm\_k} \\ E_{rel\_k} &= E_{tx\_k} + E_{rx\_k} \end{aligned} \tag{5}$$

In cluster-based WSNs, the cost functions of each node are the same as in multi-hop mesh networks except for the following two points:

(a) There is no relay energy consumption among the nodes.
(b) The cluster head has to iteratively receive the data from its leaf nodes and process the received data.

4 Classification of Task Allocation Approaches in WSNs

This section introduces the taxonomy of the task allocation approaches in WSNs. Before presenting the classification used in this chapter, some basic parameters for the classification are firstly given. In addition to the network structure and the evaluation metrics, there are also several important parameters worthy to be reported with regard to the whole task allocation procedure in WSNs, i.e., types of the tasks, optimality of the solutions and the flexibility of the approaches.

- **Types of the Tasks**
  The objective of a WSN is to monitor the physical or environmental conditions. It typically involves measuring the data, processing the measurements and communicating the results to the users. Therefore, a WSN application consists of three main types of tasks: sensing tasks, processing tasks, and communicating tasks. As widely acknowledged, the communicating tasks require the most energy consumption. This phenomena is changing recently. As more complex applications in WSNs appear, the sensing tasks and processing tasks need to be considered as well.
- **Optimality of the Solutions**
  Some approaches are able to provide the optimal solutions, while others only

can supply the suboptimal solutions. Normally, obtaining the optimal solutions increases the computational cost dramatically. The task allocation problem in WSNs has been proved to be NP-hard [15]. Thus, many works focus on the heuristic algorithms to obtain the suboptimal task allocation solutions with lower complexity. However, for the cluster-based WSNs, the number of the sensor nodes in each cluster is relatively small, which makes the optimal solutions available for each cluster. A number of studies formulate the task allocation problem based on the binary integer linear programming or linear programming to calculate the optimal solutions.

- **Flexibility of the Approaches**
  Many task allocation approaches in WSNs provide a static solution. In other words, once the approaches have been executed, the obtained solutions are used for the sensor nodes all the time. The advantage of these kinds of approaches is that the solutions are easy to implement. In contrast, the dynamic task allocation solutions are able to achieve more balanced task workload by alternatively employing multiple schedules.

The task allocation approaches in WSNs can be classified by different ways based on the above-mentioned parameters, see Table 2. By considering the effects of the network structure and the optimality of the solutions, this chapter groups the task allocation approaches in WSNs into two main categories:

(a) Optimal task allocation algorithms,
(b) Heuristic task allocation algorithms.

A systematic and detailed view of the presented approaches in each category are presented in the following sections.

5 Optimal Task Allocation Algorithms

In hierarchical cluster-based WSNs, each cluster can be optimized independently. The size of a typical cluster, i.e., the number of the leaf nodes in the cluster, is relatively small, so that the complex optimal task allocation algorithms are affordable. In this section, we firstly present the optimal algorithms that designed for only distributing sensing tasks. Then, algorithms focus on distributing not only sensing but also processing and communication tasks are introduced.

5.1 Sensing Tasks Allocation

Since quick response of a WSN could make for energy saving, a group of works focus on minimizing the execution time of the WSN sensing tasks to maximize the network lifetime [18, 19, 22, 28]. They mainly aim at eliminating transmission collisions and

Table 2 The comparison of the presented task allocation approaches used in WSNs based on effect parameters: network structure, the evaluation metrics, types of the tasks, optimality of the solutions, and flexibility of the approaches

Tasks allocation approaches		Ref. [22]	Ref. [18]	Ref. [28]	Ref. [19]	Ref. [23]	Ref. [25]	Ref. [29]	Ref. [15]	Ref. [20]	Ref. [21]	Ref. [24]	Ref. [30]	Ref. [16]	Ref. [17]
Network structures	Cluster-based WSNs	✓	✓	✓	✓	✓	✓	✓	✓						
	Multi-hop mesh WSNs									✓	✓	✓	✓	✓	✓
Metrics	Network lifetime	✓	✓	✓		✓	✓	✓	✓			✓	✓		
	Network energy cost													✓	✓
	Residual energy distribution		✓		✓					✓	✓			✓	✓
	Time requirement	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Fault tolerance														✓
Types of tasks	Sensing tasks	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Processing tasks					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Communicating tasks					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Optimality of the solutions	Optimal solution	✓	✓	✓	✓	✓	✓	✓							
	Suboptimal solution								✓	✓	✓	✓	✓	✓	✓
Flexibility of the approaches	Static task allocation	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓
	Dynamic task allocation						✓	✓							

idle gaps between two successive data transmissions for cluster-based WSNs. In the networks, the leaf nodes firstly do the sensing tasks then transmit the collected data to the cluster head; the cluster head works as a data fusion node and transmits the processed data to the sink node.

The task allocation and scheduling include two stages: intra-cluster task scheduling and inter-cluster task scheduling. The procedure for the sensing task allocation can be briefly summarized as follows: Firstly, the sink node distributes the tasks to each cluster using inter-cluster task scheduling; then, the intra-cluster task scheduling arranges the subtasks to each leaf node in the cluster. The diagram of the intra-cluster task scheduling is illustrated in Fig. 4.

Each leaf node in the cluster starts to execute the sensing task at the same time; however, they are scheduled to finish the sensing tasks and transmit the measurements orderly. For example, in Fig. 4, when the leaf node N_1 completes the transmission to the cluster head, the leaf node N_2 finishes its sensing task and starts to transmit. This process iterates until the leaf node N_K transmits the measurement to the cluster head. This iteration process in each cluster can be expressed as:

$$T_{sen\_k+1} = T_{sen\_k} + T_{cmm\_k}, \qquad k = 1, \ldots, K \tag{6}$$

where T_{sen_k} and T_{cmm_k} are the sensing time and transmitting time of leaf node N_k, respectively. Let w_{sen_k} denote the length of the sensory data (unit: bit) collected by the leaf node N_k and t_{sen_k} denote the sensing time for one bit of data. According to Eq. (3), T_{sen_k} and T_{cmm_k} are formulated as follows.

$$T_{sen\_k} = w_{sen\_k}\, t_{sen\_k}$$
$$T_{cmm\_k} = t_{o\_k} + \frac{w_{sen\_k}}{Bandwidth}$$

Thus, Eq. (6) can be rewritten as:

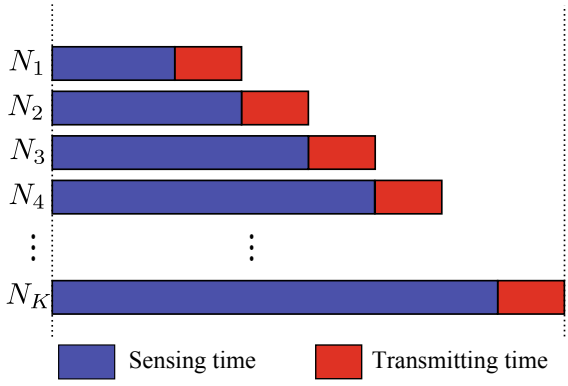


Fig. 4 Intra-cluster task scheduling: $N_1, \ldots, N_k, \ldots, N_K$ are the K leaf nodes in the cluster (Adapted from [19].)

$$w_{sen\_k+1} = w_{sen\_k}\,\alpha_k + \beta_k, \qquad k = 1, \ldots, K \tag{7}$$

where

$$\alpha_k = \frac{t_{sen\_k} + 1/Bandwidth}{t_{sen\_k+1}}, \quad \beta_k = \frac{t_{o\_k}}{t_{sen\_k+1}}$$

Once the sink node assigns the sensing task, W, to the cluster head c, it is obvious that $W = \sum_{k=1}^{K} w_{sen\_k}$. Combining Eq. (7), the sensing task w_1, assigned to the leaf node 1, can be calculated by the following equation.

$$w_{sen\_1} = \frac{W - A}{1 + B} \tag{8}$$

where

$$A = \sum_{k=2}^{K} \sum_{j=1}^{k-2} \beta_j \prod_{i=j+1}^{k-1} \alpha_i + \sum_{k=2}^{K} \beta_{k-1} \quad \text{and}$$

$$B = \sum_{k=2}^{K} \prod_{i=1}^{k-1} \alpha_i$$

For a specific sensor node, the time of sensing one bit of data t_{sen_k} is constant, it is therefore that the coefficients A and B in Eq. (8) are constants.

After calculating w_{sen_1} using Eq. (8), the cluster head computes the sensing tasks for the rest leaf nodes in the cluster based on w_{sen_1} using Eq. (7) iteratively. It is obvious that there is no idle gap between each transmission of the leaf nodes in the cluster. Moreover, the intra-cluster task scheduling also avoids the transmission conflicts among the cluster, which saves the energy consumption of the retransmission, too.

Since the cluster head acts as a data fusion node, it has to execute further processing task and then transmit the processed data to the sink node. Figure 5 shows the procedure of the inter-cluster scheduling. The inter-cluster task scheduling ensures

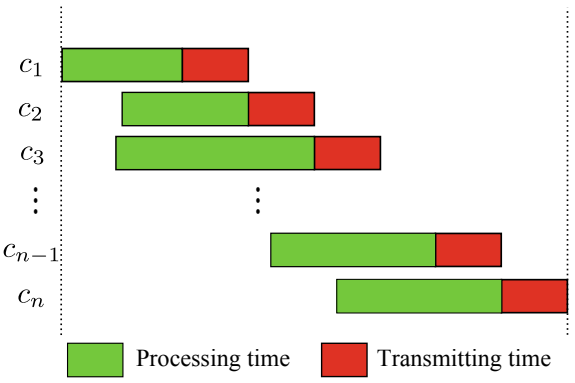


Fig. 5 Inter-cluster task scheduling: $c_1, c_2, \ldots, c_n$ represent n cluster heads (Adapted from [19].)

that each cluster receives the appropriate tasks by considering the processing and transmitting time of each cluster head. Like the intra-cluster scheduling, each cluster head finishes the data fusion task and transmits the processed data to the sink node orderly. Therefore, the idle gap and the transmission collision among each cluster head are avoided. The derivation of the calculation of the task assignment for each cluster head is not illustrated in this section, since it is very similar to the intra-cluster scheduling computation.

Consequently, by combining the intra-cluster and inter-cluster task allocation schemes, the makespan from the first sensing task being executed to the measurement of the last sensing task received by the sink node is minimized.

5.2 General Tasks Allocation

Only focusing on sensing task allocation to reduce the transmission collisions and idle gaps is insufficient for current WSN applications. The computation and communication tasks need to be considered as well. As analyzed in Sect. 3.1, the complete application including the sensing, processing and communicating activities can be modeled as a directed acyclic graph (DAG). Therefore, in cluster-based WSNs, the tasks allocation problem between the leaf nodes and cluster head can be modeled as dividing the DAG into two subgraphs with a partition solution. One subgraph is executed in the leaf node and the other one is executed by the cluster head. The partition solution can be either a static partition cut or multiple partition cuts with corresponding weights for each leaf node and the cluster head. The whole WSN application is completed by the cooperation of the leaf nodes and the cluster head, and the measurements are always transmitted from the leaf nodes to the cluster head. Therefore, the partition cut has to satisfy two *constraints*:

(1) There should be at least one task executed in either the leaf nodes or the cluster head.
(2) It has to guarantee that the data packets are transmitted from the leaf nodes to the cluster head.

The authors in [23] propose a static partition solution for task allocation based on integer linear programming (ILP). The partition cut is represented by a binary vector $\mathbb{X} = [x(v_1), \ldots, x(v_i), \ldots, x(v_{N_t})]$, where $x(v_i)$ is a boolean parameter used to indicate whether task v_i belongs to the cluster head or the leaf nodes as formulated in the following; N_t is the number of the tasks of the given DAG.

$$x(v_i) = \begin{cases} 1, & \text{if } v_i \in \text{leaf node} \\ 0, & \text{if } v_i \in \text{cluster head} \end{cases}$$

According to the cost function Eq. (2), the processing energy of the leaf node N_k and cluster head by executing the tasks, E_{p_k} and E_{p_c}, are formulated as functions of the partition cut $\mathbb{X}$.

$$
\begin{aligned}
E_{p\_k} &= \sum_{i=1}^{N_t} e_{p\_ki}\, x(v_i) = \mathbb{E}_{p\_k}\, \mathbb{X}_k^T \\
E_{p\_c} &= \sum_{i=1}^{N_t} e_{p\_ci}\, \big(1 - x(v_i)\big) = \mathbb{E}_{p\_c}\, (\mathbb{1} - \mathbb{X}_k^T)
\end{aligned}
\tag{9}
$$

where the matrices $\mathbb{E}_{p\_k} = [e_{p\_k1}, \ldots, e_{p\_kN_t}]$ and $\mathbb{E}_{p\_c} = [e_{p\_c1}, \ldots, e_{p\_cN_t}]$ represent the processing cost of each task when they are executed by leaf node N_k and the cluster head, respectively.

In a specific DAG, each task firstly consumes the received data from its predecessors, then generates and transmits to its successors. Let $\mathbb{L}_{net} = [l_{net\_1}, \ldots, l_{net\_N_t}]$ denote the net data generated by each task, i.e., the difference between its generated and received data. The amount of data transmitted from the leaf node N_k to the cluster head, L, can be expressed as:

$$
L = \sum_{i=1}^{N_t} l_{net\_i} x(v_i) = \mathbb{L}_{net} \mathbb{X}_k^T
$$

Correspondingly, the energy cost of leaf node N_k and cluster head for transmitting and receiving L bits of data, E_{tx_k} and E_{rx_c}, are:

$$
\begin{aligned}
E_{tx\_k} &= e_{o\_k} + e_{tx\_k}\, L = e_{o\_k} + e_{tx\_k}\, \mathbb{L}_{net}\, \mathbb{X}_k^T \\
E_{rx\_c} &= e_{o\_c} + e_{rx\_c}\, L = e_{o\_c} + e_{rx\_c}\, \mathbb{L}_{net}\, \mathbb{X}_k^T
\end{aligned}
\tag{10}
$$

As analyzed in Sect. 3.3, the cluster head has to iteratively receive the data from its leaf nodes and process the received data. The complete energy consumption of leaf node N_k and the cluster head, E_k and E_c, are:

$$
\begin{aligned}
E_k(\mathbb{X}_k) &= E_{p\_k} + E_{tx\_k} \\
&= e_{o\_k} + (\mathbb{E}_{p\_k} + e_{tx\_k}\, \mathbb{L}_{net})\, \mathbb{X}_k^T \\
E_c(\mathbb{X}_{1,\ldots,K}) &= \sum_{k=1}^{K} (E_{p\_c} + E_{rx\_c}) \\
&= \sum_{k=1}^{K} \big(e_{o\_c} + \mathbb{E}_{p\_c}\, \mathbb{1} + (e_{rx\_c}\, \mathbb{L}_{net} - \mathbb{E}_{p\_c})\, \mathbb{X}_k^T\big)
\end{aligned}
\tag{11}
$$

Note that all of the parameters in Eq. (11) are constant except the partition cut $\mathbb{X}$.

The problem of maximizing the network lifetime is equivalent to minimizing its reciprocal. When the network lifetime is defined as the time when the first node dies, its maximization can be expressed as a binary ILP:

$$\begin{aligned} &\arg\min_{\mathbb{X}_k} \max \left\{ \frac{E_c(\mathbb{X}_{1,\ldots,K})}{B_c}, \frac{(E_1(\mathbb{X}_1)}{B_1}, \ldots, \frac{E_K(\mathbb{X}_K)}{B_K} \right\} \\ &\text{subject to :} \\ &\qquad constraint\,(1) \quad 1 \leq \mathbb{1}\, \mathbb{X}_k^T \leq N_t - 1 \\ &\qquad constraint\,(2) \quad \mathbb{B}^T\, \mathbb{X}_k^T \leqq \mathbb{0} \end{aligned} \tag{12}$$

where B_c and $B_1, \ldots, B_K$ are the battery of the cluster head and the K leaf nodes, respectively; $\mathbb{0}$ is an all zero vector; $\mathbb{B}$ is the incidence matrix of the DAG, it has the row for each vertex and column for each edge: $\mathbb{B}(v, e)$ equals 1 if edge e leaves vertex v, -1 if edge e enters v and 0 otherwise.

By solving the above binary ILP, i.e., Eq. (12), the optimal static partition solution can be calculated and exploited to maximize the network lifetime.

Instead of using the static partition solution, the use of multiple partition cuts can achieve a more balanced task allocation. Let us take the following simple scenario as an example: There are only one leaf node and one cluster head with the same battery energy in the network. The tasks of the application are modeled as shown in Fig. 6, executing task v_2 costs much more energy than executing task v_1 and v_3. The static partition solution using either $\mathbb{X}_1$ or $\mathbb{X}_2$ will lead to a fast death of the cluster head or the leaf node, respectively. While the energy consumption of the leaf node and cluster head can be more balanced by using both the partitions iteratively.

To this end, the work [25] extends [23] and proposes a dynamic task allocation algorithm using multiple partition cuts. It formulates the task allocation problem as a linear programming based on the average energy consumption of the leaf node and cluster head, $\overline{E}_k$ and $\overline{E}_c$. By assuming the network collapses after J rounds, $\overline{E}_k$ and $\overline{E}_c$ can be formulated as:

$$\begin{aligned} \overline{E}_k(\mathbb{\Gamma}_k) &= \frac{1}{J} \sum_{j=1}^{J} E_k^j(\mathbb{X}_k^j) \\ &= e_{o\_k} + (\mathbb{E}_{p\_k} + e_{tx\_k}\, \mathbb{L}_{net})\, \mathbb{\Gamma}_k^T \\ \overline{E}_c(\mathbb{\Gamma}_{1,\ldots,K}) &= \frac{1}{J} \sum_{j=1}^{J} E_c^j(\mathbb{X}_{1,\ldots,K}^j) \\ &= \sum_{k=1}^{K} \left(e_{o\_c} + \mathbb{E}_{p\_c} \mathbb{1} + (e_{rx\_c} \mathbb{L}_{net} - \mathbb{E}_{p\_c}) \mathbb{\Gamma}_k^T \right) \end{aligned} \tag{13}$$

where $E_k^j(\mathbb{X}_k^j)$ is the energy cost of leaf node N_k when it applying the partition cut $\mathbb{X}_k^j$ in the jth round; $E_c^j(\mathbb{X}_{1,\ldots,K}^j)$ is the energy cost of the cluster head in the jth round; $\mathbb{\Gamma}_k = \frac{1}{J} \sum_{j=1}^{J} \mathbb{X}_k^j = [\gamma_k(v_1), \ldots, \gamma_k(v_i), \ldots, \gamma_k(v_{N_t})]$, and each element $\gamma_k(v_i) = \frac{1}{J} \sum_{j=1}^{J} x_k^j(v_i)$ satisfies $0 \leq \gamma_k(v_i) \leq 1$. Thus, $\gamma_k(v_i)$ can be treated as the probability of how often the task v_i is allocated to the leaf node N_k.

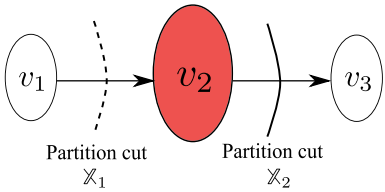


Fig. 6 An example of the dynamic scheduling scheme with 2 partition cuts

As the network lifetime is also defined as the time from the start of the network until the first sensor node runs out of battery. The Network lifetime, NL, can be expressed as:

$$NL = \min\left\{\frac{B_c}{\overline{E}_c(\mathbb{\Gamma}_{1,\ldots,K})}, \frac{B_1}{\overline{E}_1(\mathbb{\Gamma}_1)}, \ldots, \frac{B_K}{\overline{E}_K(\mathbb{\Gamma}_K)}\right\} \tag{14}$$

The maximization of Eq. (14) can be formulated as a standard linear programming problem as follows:

$$\begin{aligned} &\arg\min_{\mathbb{\Gamma}_k} \max\left\{\frac{\overline{E}_c(\mathbb{\Gamma}_{1,\ldots,K})}{B_c}, \frac{\overline{E}_1(\mathbb{\Gamma}_1)}{B_1}, \ldots, \frac{\overline{E}_K(\mathbb{\Gamma}_K)}{B_K}\right\} \\ &\text{subject to :} \\ &\qquad \textit{constraint}\ (1) \quad 1 \leq \mathbb{1}\,\mathbb{\Gamma}_k^T \leq N_t - 1 \\ &\qquad \textit{constraint}\ (2) \quad \mathbb{B}^T\mathbb{\Gamma}_k^T \leqq \mathbb{0} \end{aligned} \tag{15}$$

The optimal solution of the dynamic task allocation can be calculated by solving the above LP. After obtaining $\mathbb{\Gamma}_k$, the corresponding partition cuts and the weights can be easily calculated. Taking the DAG in Fig. 6, for example, assuming the solution $\mathbb{\Gamma}_{\mathbb{k}} = [1\ 0.6\ 0] = 0.4[1\ 0\ 0] + 0.6[1\ 1\ 0]$, it indicates that the partition cuts $\mathbb{X}_1$ and $\mathbb{X}_2$ should be executed 40 and 60% of the time by the leaf node and cluster head to reach the maximum lifetime.

The performance of the above introduced optimal static and dynamic task allocation algorithms, in terms of the network lifetime increase for executing MEPS and spectrum applications,[3] is presented in Figs. 7 and 8, respectively. Compared with the no task allocation strategy, in which the leaf nodes directly transmit the measurement and the cluster head executes all of the rest tasks, the network lifetime is extended dramatically by task allocation approaches. Moreover, the dynamic task allocation using multiple partition cuts performs better. Although the above-mentioned algorithms need to be executed off-line, they must know the parameters of the nodes and the networks in advance.

[3]MEPS is the maximum entropy power spectrum (MEPS) computation which is adapted from Ptolemy II design environment and the spectrum computation refers to convert signals from time domain to frequency domain.

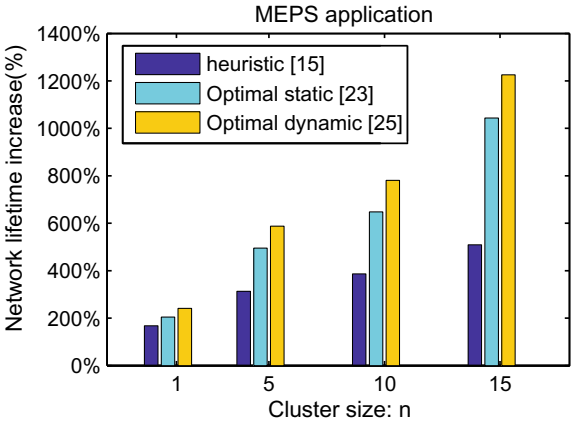


Fig. 7 The network lifetime improvements by using the optimal static [23] and dynamic [25], and heuristic [15] task allocation algorithms with respect to using no task allocation algorithm when executing the MEPS application (Redraw by combing [15, 23, 25].)

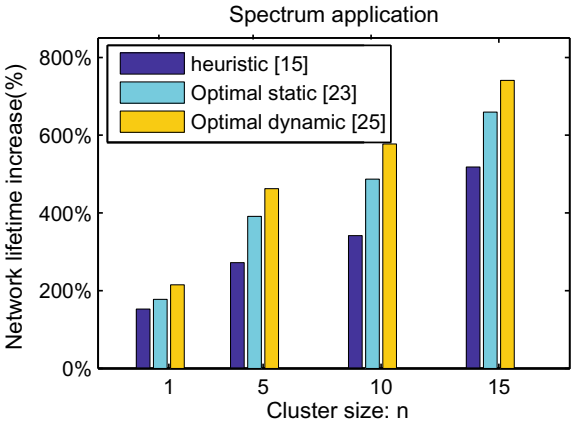


Fig. 8 The network lifetime improvements by using the optimal static [23] and dynamic [25], and heuristic [15] task allocation algorithms with respect to using no task allocation algorithm when executing the spectrum application (Redraw by combing [15, 23, 25].)

5.3 Online Distributed Task Allocation

In realistic scenarios, it is hard or even impossible to obtain the detailed network parameters in advance. Moreover, the networks are not always static: The network size and the positions of the nodes may change over time. Off-line methods are not efficient enough to maximize the network lifetime; even worse, they may decrease the network lifetime.

To this end, a report, [29], presents an in-depth analysis of the optimal task allocation solution and then proposes an online optimal distributed task allocation algorithm for cluster-based WSNs.

5.3.1 Composition of the Optimal Partition Solution

Although the work [25] claims that the optimal task partition solution for each leaf node always consists of two partition cuts with the corresponding weights, it only supplies simulation results. Based on this phenomenon, an in-depth theoretic analysis is provided by [29].

Given a specific DAG graph, since the total number of the tasks is not infinite, the number of the valid partition cuts is limited. According to the cost functions in Sect. 3.3, each partition cut can be associated with a point, (E_k, E_{ck}), on the energy plane, which represents the energy consumption of leaf node N_k and the cluster head, E_k, and E_{ck}. At the same time, the execution time of leaf node i and the cluster head when exploiting each partition cut can be calculated. Given the thresholds of the time requirement (depends on specific WSN applications), those points which cannot satisfy the time constraint are useless and will be removed. The corresponding energy points of the rest partition cuts and their combinations construct a convex set as shown in Fig. 9.

The smaller the energy consumption of the node, the longer it can survive. Thus, maximizing the network lifetime is equivalent to find the points that minimize the energy consumption of leaf node N_k under the same energy consumption of the cluster head, and vice versa. These points correspond to the subset of the boundary of the convex set, which is called *optimal curve* in Fig. 9. For any point in the convex set, there always exists at least one point on the *optimal curve* that both E_{ck} and E_k are smaller. It means that the optimal partition solution for leaf node N_k is definitely on the *optimal curve*. This line contains some points (*important partition cuts*) whose linear combination generates all the optimal solutions. Thus, the optimal solution in

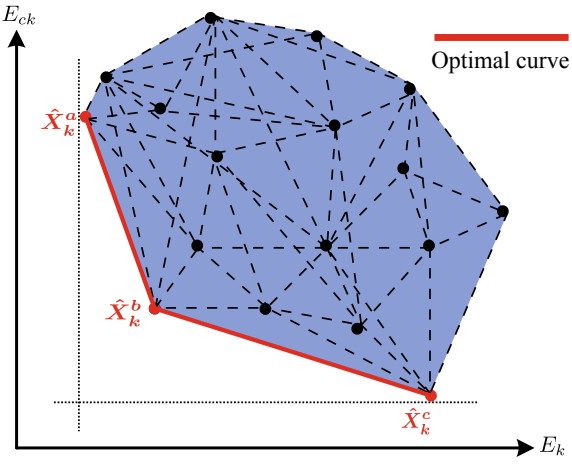


Fig. 9 All combinations of the valid partition cuts and the corresponding energy consumption of leaf node N_k and the cluster head

this example is either one of the *important partition cuts*, $\hat{\boldsymbol{X}}_k^a$, $\hat{\boldsymbol{X}}_k^b$ and $\hat{\boldsymbol{X}}_k^c$, or the linear combinations of $\hat{\boldsymbol{X}}_k^a$ and $\hat{\boldsymbol{X}}_k^b$ or $\hat{\boldsymbol{X}}_k^b$ and $\hat{\boldsymbol{X}}_k^c$.[4]

To maximize the network lifetime, it only needs to keep the *important partition cuts*, while others can be neglected. This lays the foundation for the proposed online distributed scheduling algorithm.

5.3.2 Online Distributed Scheduling Algorithm

The online distributed workload scheduling algorithm is actually an online negotiation among the leaf nodes and cluster head. It is necessary to analyze and formulate the energy consumption of the *optimal curve* before presenting the online algorithm.

The DAG graphs of the applications can be easily modeled before deploying the networks in realistic scenarios. The *important partition cuts* and the *optimal curve* of leaf node N_k can be obtained off-line based on the binary decision diagram. For the detailed computation information, please refer to [29]. The relation between E_{ck} and E_k on this *optimal curve* can be formulated as Eq. (16), which will be stored in leaf node N_k before the network is implemented.

$$E_{ck} = \begin{cases} A_0 E_k + B_0, & \text{if } E_k \in \left[E_k(\hat{\boldsymbol{X}}_k^a), E_k(\hat{\boldsymbol{X}}_k^b)\right] \\ A_1 E_k + B_1, & \text{if } E_k \in \left[E_k(\hat{\boldsymbol{X}}_k^b), E_k(\hat{\boldsymbol{X}}_k^c)\right] \end{cases} \tag{16}$$

where

$$A_0 = \frac{E_c(\hat{\boldsymbol{X}}_k^a) - E_c(\hat{\boldsymbol{X}}_k^b)}{E_k(\hat{\boldsymbol{X}}_k^a) - E_k(\hat{\boldsymbol{X}}_k^b)}, A_1 = \frac{E_c(\hat{\boldsymbol{X}}_k^b) - E_c(\hat{\boldsymbol{X}}_k^c)}{E_k(\hat{\boldsymbol{X}}_k^b) - E_k(\hat{\boldsymbol{X}}_k^c)},$$

$$B_0 = \frac{E_k(\hat{\boldsymbol{X}}_k^a)E_c(\hat{\boldsymbol{X}}_k^b) - E_c(\hat{\boldsymbol{X}}_k^a)E_k(\hat{\boldsymbol{X}}_k^b)}{E_k(\hat{\boldsymbol{X}}_k^a) - E_k(\hat{\boldsymbol{X}}_k^b)} \quad \text{and}$$

$$B_1 = \frac{E_k(\hat{\boldsymbol{X}}_k^b)E_c(\hat{\boldsymbol{X}}_k^c) - E_c(\hat{\boldsymbol{X}}_k^b)E_k(\hat{\boldsymbol{X}}_k^c)}{E_k(\hat{\boldsymbol{X}}_k^b) - E_k(\hat{\boldsymbol{X}}_k^c)}.$$

It is a continuous and monotonically decreasing function.

After the off-line preparation, the online negotiation is started among the leaf nodes and cluster head to obtain the maximum network lifetime. To clearly illustrate the internal procedure of the online algorithm, a naive method is presented firstly. When the network starts to work, the cluster head firstly broadcasts an expected network lifetime T_e. Each leaf node computes its own expected energy consumption $E_{e\_k} = B_k / T_e$ and the corresponding E_{e_ck} according to (16), and then transmits them to the cluster head. After receiving the messages from all leaf nodes, the cluster head examines whether its battery energy is enough to last T_e time. If it still has

[4] The important partition cuts are represented by the binary vectors as the same as the partition cut $\mathbb{X}$ in Sect. 5.2.

residual battery energy, i.e., $B_c > T_e \sum_{k=1}^{K} E_{e\_ck}$, it broadcasts a larger T_e, otherwise a smaller one. The leaf nodes calculate the corresponding E_{e_k} and E_{e_ck} again until $B_c = T_e \sum_{k=1}^{K} E_{e\_ck}$. At last, the cluster head broadcasts one confirm message, and the leaf nodes individually calculate their own partition solutions based on the final E_{e_k}. Although this naive method is simple, it may need a large quantity of message exchange, which results in too much energy consumption.

Fortunately, [29] has proved that when the lifetimes of the leaf nodes and cluster head equal each other i.e., $T_1 = \cdots = T_K = T_c$, the maximum network lifetime is achieved. Based on this proof, [29] proposes the online optimal distributed task allocation algorithm with dramatically reduced iterations.

Unlike the naive method where the cluster head randomly adjusts T_e according to the received messages, the improved algorithm enables the cluster head to calculate the temporary optimal network lifetime $\hat{T}$. After each leaf node receives T_e, it transmits not only its expected energy cost $E_{e\_k} = B_k / T_e$ and the corresponding E_{e_ck} calculated by Eq. (16), but also the slope, k_{e_k}, of the segment which includes the point (E_{e_k}, E_{e_ck}) as shown in Fig. 10. Let $(\hat{E}_k, \hat{E}_{ck})$ represent the energy point corresponding to the optimal network lifetime $\hat{T}$. Assuming it is still on the current segment, $\hat{E}_{ck}$ can be calculated by:

$$\hat{E}_{ck} = E_{e\_ck} + k_{e\_k}(\hat{E}_k - E_{e\_k}) \tag{17}$$

According to proof that the maximum network lifetime achieves when the lifetimes of the leaf nodes and cluster head equal each other, there exists:

$$\hat{T} = \frac{B_k}{\hat{E}_k} = \frac{B_c}{\sum_{k=1}^{K} \hat{E}_{ck}} \tag{18}$$

Since $E_{e\_k} = B_k / T_e$, combining (17) and (18), $\hat{T}$ can be calculated in the cluster head by:

$$\hat{T} = \frac{B_c - T_e \sum_{k=1}^{K} k_{e\_k} E_{e\_k}}{\sum_{k=1}^{K} E_{e\_ck} - k_{e\_k} E_{e\_k}} \tag{19}$$

Then, the cluster head compares $\hat{T}$ with T_e. If they are different, it broadcasts the current expected network lifetime $T_e = \hat{T}$. Leaf node N_k repeats the calculation of E_{e_k}, E_{e_ck} and k_{e_k}, and sends them to the cluster head. Once $\hat{T}$ equals T_e, the cluster head broadcasts a $confirm$ message. The last received T_e is actually the final maximum network lifetime; leaf node N_k can easily calculate its own optimal partition cuts and the weights based on it. For instance, assuming $\hat{T}$ in Fig. 10 is the final maximum lifetime, the optimal partition solution for leaf node i consists of two partition cuts, $\hat{\boldsymbol{X}}_{\boldsymbol{k}}^{\boldsymbol{b}}$ and $\hat{\boldsymbol{X}}_{\boldsymbol{k}}^{\boldsymbol{c}}$, with the related weights ω_b and ω_c.

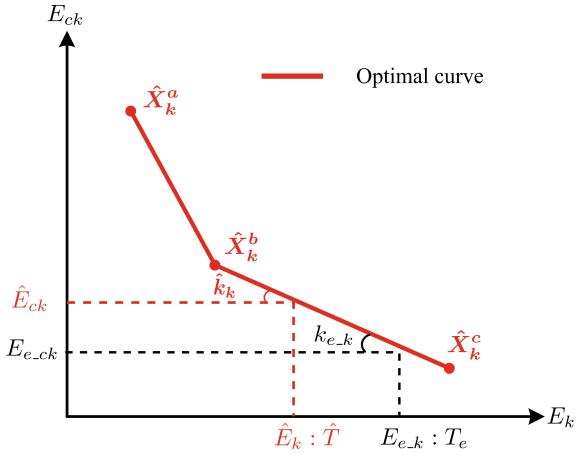


Fig. 10 The calculation of the temporary optimal lifetime based on the *important partition cuts*

$$\omega_b = \frac{E_k(\hat{\boldsymbol{X}}_{\boldsymbol{k}}^{\boldsymbol{c}}) - \hat{E}_k}{E_k(\hat{\boldsymbol{X}}_{\boldsymbol{k}}^{\boldsymbol{c}}) - E_k(\hat{\boldsymbol{X}}_{\boldsymbol{k}}^{\boldsymbol{b}})}$$
$$\omega_c = \frac{E_k(\hat{\boldsymbol{X}}_{\boldsymbol{k}}^{\boldsymbol{b}}) - \hat{E}_k}{E_k(\hat{\boldsymbol{X}}_{\boldsymbol{k}}^{\boldsymbol{b}}) - E_k(\hat{\boldsymbol{X}}_{\boldsymbol{k}}^{\boldsymbol{c}})} \tag{20}$$

The pseudocodes that executed in the cluster head and leaf node N_k are shown in Algorithm 1 and Algorithm 2. The simulation result of the network lifetime increase when applying the online distributed algorithm [29] is demonstrated in Fig. 11. It is obvious that the online distributed algorithm [29] achieves the same performance as the off-line dynamic task allocation algorithm [25] in terms of the network lifetime increase. Moreover, since it is executed online, it can flexibly address the dynamic changes of the networks.

Algorithm 1 Cluster head algorithm

1: Initialize T_e and broadcast it
2: **for** each calculation round **do**
3: Receive E_{e_ck}, k_{e_k} and E_{e_k}
4: Calculate $\hat{T}$ using Eq. (19)
5: **if** $\hat{T} == T_e$ **then**
6: Broadcast $confirm$
7: **Break**
8: **else**
9: $T_e = \hat{T}$, and broadcast T_e
10: **end if**
11: **end for**

Algorithm 2 Leaf node algorithm

1: **for** each received message **do**
2: **if** not $confirm$ **then**
3: Calculate E_{e_k}, E_{e_ck} and k_{e_k}, using Eq. (16), and transmit them
4: **else**
5: $\hat{E}_k = B_k / T_e$
6: Calculate partition weights using Eq. (20)
7: **Break**
8: **end if**
9: **end for**

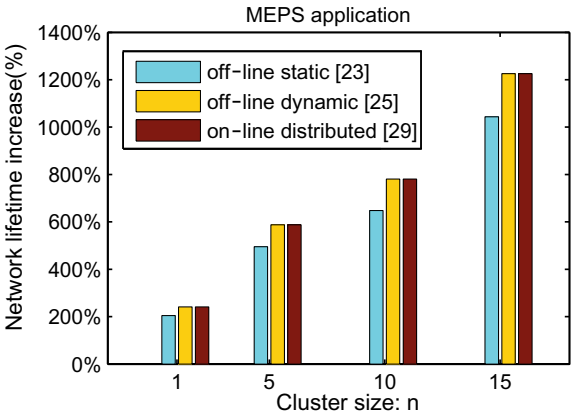


Fig. 11 The network lifetime improvements by using the optimal static [23] and dynamic [25], and online distributed [29] task allocation algorithms with respect to using no task allocation algorithm when executing the MEPS application (Redraw by combing [23, 25, 29].)

6 Heuristic Task Allocation Algorithms

As the number of nodes in the network increases, the complexity of the optimal task allocation algorithms may get too large. Especially for multi-hop WSNs, executing the optimal task allocation is too complex. Many heuristic algorithms are designed to reduce the complexity while providing a good suboptimal solution or an optimal task allocation solution with a given probability.

The category of the heuristic task allocation algorithms could be generally classified into two main groups: the traditional heuristic and the bio-inspired heuristic algorithms.

6.1 Traditional Heuristic Algorithms

Traditional heuristic algorithms mainly reduce the complexity by seeking the local optimal solutions. Since the application tasks can be modeled as a DAG, a variety of existing heuristic algorithms for graph partitioning are modified for WSNs. The authors in [15] adopt the ideas from the Kernighan–Lin (K-L) [31] and Fiduccia–

Mattheyses (F-M) [32] algorithms and develop a heuristic approach for task allocation. The task allocation problem is defined as the energy-driven partitioning (EDP) problem, while considering the energy costs associated with computation and communication. As it is also designed for cluster-based WSNs, the EDP problem is also modeled as dividing the DAG $G = (V, E)$ into two subgraphs, G_{leaf} and G_{head} with a partition cut. Moreover, the partition cut cannot violate the *constraint* (1) and *constraint* (2) as analyzed in Sect. 5.2. In the paper, the authors consider that the leaf nodes have the same the energy parameters and share one partition cut with the cluster head. A cost function of a partition cut, CT, is defined as:

$$CT = max\{E_{leaf}, E_{head}\}$$

where E_{leaf} and E_{head} are the energy cost of each leaf node and the cluster head when applying one partition cut. By assuming that the battery energy of the leaf nodes and the cluster head are equal, the problem of maximizing the network lifetime is formulated as $min\{CT\}$.

Given an initial partition cut, the main idea of the heuristic algorithm is to serially switch the task between G_{leaf} and G_{head} according to their gain function $\delta(v)$. $\delta(v)$ is the energy reduction or increase of CT when moving the task v from one of the subgraphs to the other, which can be calculated efficiently based on the cost functions as analyzed in Sect. 3.3. The pseudocode of the heuristic algorithm is illustrated in Algorithm 3, and the performance on network lifetime increase and algorithm executing time are depicted in Figs. 7, 8 and 12, respectively.

It is easy to realize that the heuristic task allocation algorithm needs less computation time while providing shorter network lifetime extension compared with the optimal approaches. Surprisingly, the execution time of the binary ILP-based optimal algorithm [23] is just slightly larger, which indicates the feasibility of the optimal algorithms in cluster-based WSNs.

In the last few years, a number of new heuristic task allocation algorithms have been proposed based on game theory [20, 21]. In such works, the task allocation

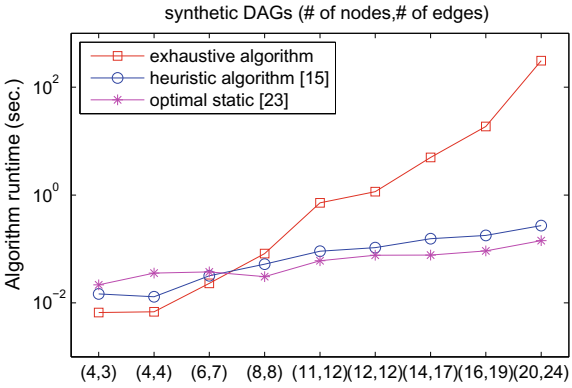


Fig. 12 The algorithm runtime comparison of the heuristic [15], optimal static [23] and the exhaustive task allocation algorithms when executing the synthetic DAGs (Redraw from [23])

Algorithm 3 Heuristic task allocation algorithm [15]

Input: a DAG ($G = (V, E)$)
Output: subgraphs G_{leaf} and G_{head}

```
Create an initial partition and get the initial G_leaf and G_head
Copy them: G̃_leaf = G_leaf, G̃_head = G_head
Computer CT for the initial partition
while not break do
    for each task v ∈ V do
        if v is unlocked and satisfies constraints 1) and 2) then
            Mark v free, and compute δ(v)
        end if
    end for
    if free tasks exist then
        Find task v with the maximum δ(v)
        Add (v, δ(v)) to the order list: order_L
        Update G̃_leaf and G̃_head by switching v
        CT = CT − δ(v), and mark v locked
    else
        Break
    end if
end while
Select first k tasks that maximizes ∑_v δ(v) from order_L
if ∑_v δ(v) ≥ 0 then
    Update G_leaf and G_head by switching the k tasks
end if
```

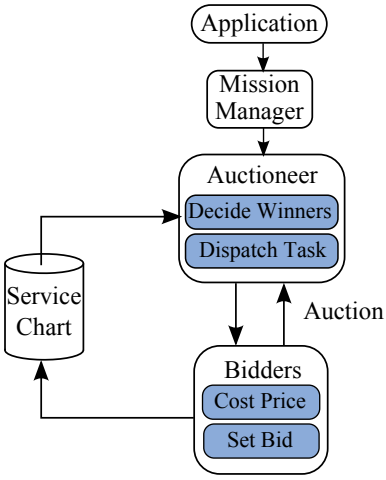


Fig. 13 The market-based architecture for task allocation in WSNs (Modified from [20, 21].)

is modeled as a market-based architecture which mainly includes *mission manager*, *auctioneer*,*bidders*, and *service chart* as depicted in Fig. 13.

Given a specific application, the *mission manager* firstly formulates the application as a DAG and then delivers the tasks to the *auctioneer* according to their priorities. When the *auctioneer* receives the task, it transmits a *task message*, which is made up

of task size and task deadline, to the *bidders*. Typically, the sink node plays the role of the *auctioneer*. Note that it is possible to have more than one auctioneers for multiple applications. The *bidders* are the sensor nodes. After each *bidder* receives one task message from the *auctioneer*, they calculate their cost price for accomplishing the task based on the task processing energy consumption, the involved communication cost, the task deadline and their own residual battery energy. Then, the *bidders* send their bids to the *auctioneer*. According to the cost prices, the *auctioneer* makes its decision to choose the winner and assigns this task to it. The cost price of the winning *bidder* is recorded in the *service chart*, which will be accumulated to the winning *bidder* when it calculates the cost price of the next task message.

To reduce the communication overhead of the message exchanging among the *auctioneer* and the *bidders*, the work [20] proposes a distributed message exchange method. When receiving the broadcast of the task message from the *auctioneer*, each sensor node sets a waiting time proportional to its own cost price and goes to a LISTEN mode. Therefore, the winner wakes up earlier than others and broadcasts its bid. If other nodes receive the winner's bid, they automatically drop the current competition do not transmit to the *auctioneer*. The performance of game theory-based task allocation algorithm is shown in Fig. 14.

Although the game theory-based task allocation algorithms are able to balance the energy consumption of the network, the efficiency of this kind of algorithms in large networks still needs to be validated. Moreover, getting stuck in the local optimum is a common drawback for most of the traditional task allocation algorithms.

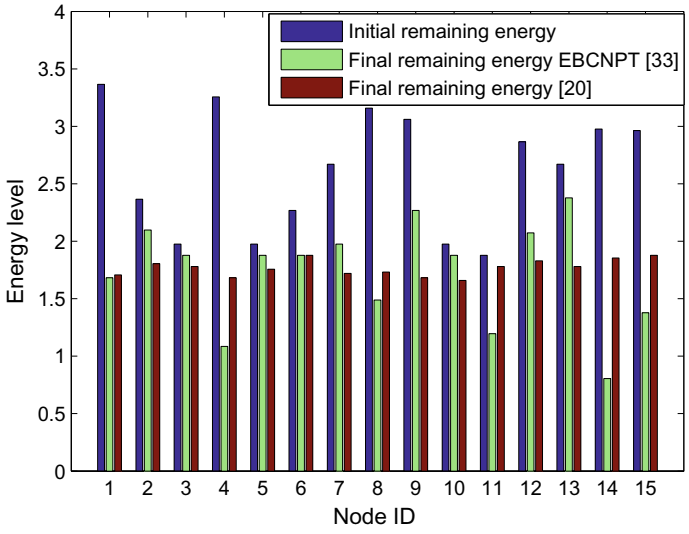


Fig. 14 The remaining energy comparison after allocating 50 tasks to 15 nodes by using the game theory-based task allocation algorithm [20] and a modified energy-balanced critical node path tree algorithm (EBCNPT) [33] (Regenerated from [20].)

6.2 Bio-inspired Heuristic Algorithms

Compared with the traditional heuristic algorithms which usually get stuck in local optimum, the bio-inspired heuristic algorithms adopting the bionic intelligence can obtain the global optimal solutions with high probability. Among the current studies, genetic algorithm (GA) [24, 30] and particle swarm optimization (PSO) [16, 17, 34, 35] are widely employed.

GAs are the heuristic search and optimization techniques that imitate the biological process of natural evolution. They are used to search the "fittest" solutions for a given maximization or minimization problem. Almost all genetic algorithms have at least the following common components: genetic representations of the solution domains (generations of chromosomes), a fitness function for optimization, selection according to fitness, crossover to produce new generation, and random mutation in new generation [36].

The authors in [24, 30] adopt GAs to address the task allocation problems in multi-hop wireless sensor networks. A complete task allocation solution is modeled as a chromosome, C, which is encoded as a 3-by-N_t matrix, where N_t is the number of the tasks in the DAG. The elements in the first row of C represent the tasks. The corresponding places in the second and third rows stand for the nodes ID and the processing workloads of tasks. In addition, the communication energy cost of the edges, TR, is encoded as a 3-by-N_e matrix, where N_e is the number of the edges in the DAG. The three elements of each column in the TR matrix are the sender task, the receiver task and the communication data packets, respectively. In multi-hop networks, the sender task and the receiver task could be allocated to nodes that are several hops away from each other. Therefore, the relay energy cost should be considered for the nodes who act as the route nodes. Figure 15 illustrates how to adapt the chromosome and communication energy cost matrices to multi-hop networks by adding virtual relay tasks to the DAG. Note that the relay tasks have no processing workload. Please note that GAs do not guarantee the global optimum, since the crossover and mutation processes may destroy the best solution of the generation. However, if both the parents have good genes, there is higher probability of generating better offspring chromosomes.

A hybrid fitness function is designed to perform the optimization while satisfy the latency requirements as formulated in Eq. (21).

$$
\begin{aligned}
fitness(C_i) &= \frac{NL(C_i)}{MAX\_NL} - \beta_1 \frac{\beta_2\, SL(C_i)}{MAX\_SL} \\
\beta_2 &= \begin{cases} 0, & SL(C_i) \le Deadline \\ 1, & SL(C_i) > Deadline \end{cases}
\end{aligned}
\tag{21}
$$

$NL(C_i)$ is the network lifetime when using the chromosome C_i. Note that the network lifetime is also denoted as the time when the first node dies. $SL(C_i)$ represents the schedule length of one round execution makespan time of C_i. $Deadline$ is the

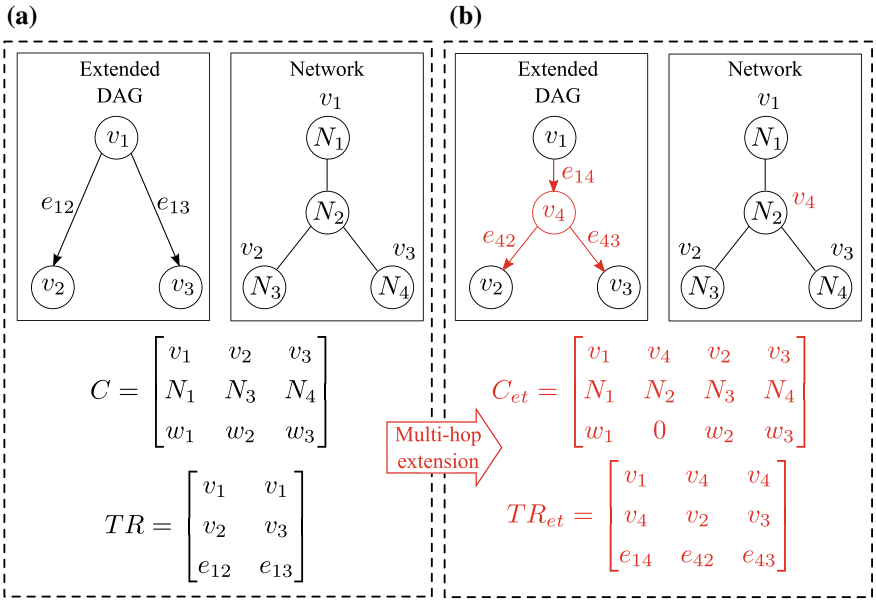


Fig. 15 An example of chromosome and communication cost in multi-hop network (Modified from [30].)

application latency requirement. β_1 is a tuning parameter decided as the maximum value that guarantees $fitness(C_i) \geq 0$.

A slight difference from general GAs, an inheritance progress is used before the selection. The top m percent of the chromosomes in the current generation are inherited to the next generation to keep the good genes. The rest is selected as pairs of chromosomes by using Roulette-Wheel scheme [37], which makes chromosomes with better fitness values have higher probability to be selected as pairs. Then, a random single point crossover method is applied. Specifically, the first and third rows of the chromosomes are fixed, while the nodes ID in the second row is switched over after the crossover point. Although the selection and crossover keep the genetic information of fitter chromosomes, they decline the genetic diversity which may easily lead the GAs stuck at a local optimum. Therefore, the mutation is used to help protect against this problem. There are two types of mutations employed in [24, 30]:

(1) Each chromosome has a probability of ϕ (mutation rate) to randomly change an allocated task to another node.
(2) Each chromosome has a probability of ϕ being completely replaced by a randomly created chromosome.

The complete GAs start with a randomly created generation of chromosomes. Then, the multi-hop extension process is executed to take the relay energy cost into account. According to cost functions in Sect. 3.3, the fitness values are calculated and ranked. The top $m\%$ of the chromosomes are inherited, while the rest generate their

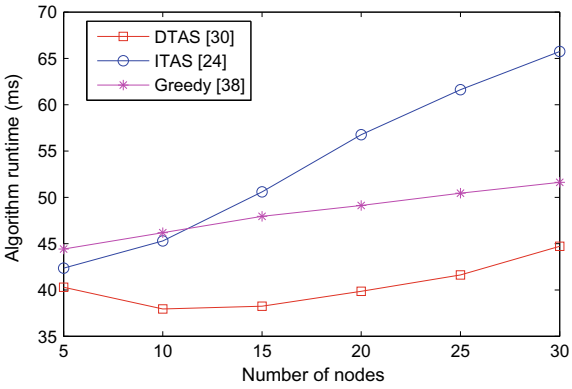


Fig. 16 The performance evaluations on algorithm runtime of ITAS [24], DTAS [30] and a greedy task allocation algorithm [38] (Recreated according to [30].)

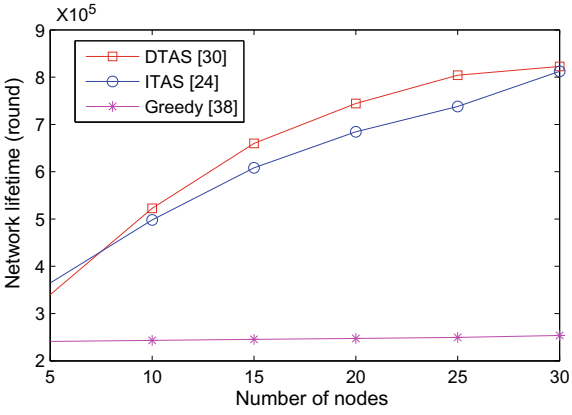


Fig. 17 The performance evaluations on network lifetime of ITAS [24], DTAS [30] and a greedy task allocation algorithm [38] (Recreated according to [30].)

offspring via selection, crossover and mutation processes. The inherited offspring and the generated offspring constitute the next generation of chromosomes. The algorithm repeats itself until the predefined maximum iterations or if the best results are achieved. Finally, the chromosome with the best fitness value in the last generation is chosen as the final task allocation solution. The performance evaluations of GAs on the execution time and network lifetime are depicted in Figs. 16 and 17, respectively.

PSO is another popular heuristic optimization algorithm which is inspired by social behavior and movement dynamics of some animals such as the flocking behavior of birds and the schooling behavior of fish. The general idea of PSO is that a population of particles (candidate solutions) repeatedly move to the next positions (x, modified solutions), according to the guides of each one's personal best solution (p), the global best solution (g) among the population and the current movements of the particles (velocity v), until g is discovered. Like GAs, a fitness function is needed in PSO for the optimization to update the better solutions.

The task allocation problem in WSNs can be described as a 0–1 decision problem where each boolean parameter indicates whether the node is selected for a task. There-

fore, binary PSO (BPSO) and its modified versions have been commonly employed [16, 17, 34, 35]. A binary string is used to represent each individual particle. The position of each bit of the particle is updated by switching the value between 0 and 1 according to the normalized probability values of their velocities. Specifically, for the d-th bit of the i-th particle in the j-th iteration, its position, x_{id}^{j}, can be expressed as:

$$x_{id}^{j} = \begin{cases} 0, & \text{if } rand() \geq S(v_{id}^{j}) \\ 1, & \text{if } rand() < S(v_{id}^{j}) \end{cases} \tag{22}$$

where $rand()$ is a random value uniformly distributed within the interval [0, 1], $S(v_{id}^{j})$ is a sigmoid transfer function that converts the real valued velocities to probabilities in [0, 1], which is formulated as:

$$S(v_{id}^{j}) = \frac{1}{1 + \exp(-v_{id}^{j})} \tag{23}$$

where v_{id}^{j} is the velocity of the dth bit; it is updated by:

$$v_{id}^{j} = w\, v_{id}^{j-1} + c_1\, r_1 (p_{id}^{j-1} - x_{id}^{j-1}) + c_2\, r_2 (g_{d}^{j-1} - x_{id}^{j-1}) \tag{24}$$

where w is the inertia weight and satisfy $0 < w \leq 1$, c_1 and c_2 are the acceleration coefficients, r_1 and r_2 are random variables which follow the uniform distribution within [0, 1], p_{id}^{j-1} and g_{d}^{j-1} are the dth elements of the personal best position of the ith particle, $\mathbb{P}_{i}^{j-1}$, and the global best position, $\mathbb{G}^{j-1}$, in $j-1$th iteration, respectively. The pseudocode of the BPSO-based task allocation algorithms used in WSNs is shown in Algorithm 4.

The fitness function is usually formulated based on the evaluation metric of the network lifetime. However, the work [16] considers the residual energy distribution (D) as a evaluation metric as well as the total task execution time (T) and the total energy cost (E). Thus, a multi-objective fitness function is designed as follows:

$$f = W_1\, T + W_2\, E + W_3\, D$$

where W_1, W_2, and W_3 are the weights of T, E, and D, respectively. They satisfy: $W_1 + W_2 + W_3 = 1$, and the values can be adjusted according to importances of the three metrics. A similar multi-objective fitness function is also proposed in [17] by considering fault-tolerant requirements.

Standard BPSO suffers trapping in the local optimal and hardly converging to the global optimum [34]. By analyzing the effect of the inertia weight parameter on BPSO, the authors in [35] propose an adaptive inertia weight scheme. It enables BPSO gradually coverage by linearly increasing the inertia weight. The work [16] adopts the idea of mutation operation in GAs to keep the diversity of the particles and reduce the probability of being stuck in local optimum. After using Eq. (22) to update the position, each undated position has the probability of ϕ being placed by

Algorithm 4 BPSO-based task allocation algorithm

1: Initialize the WSN, the application tasks
2: Initialize the population of particles: positions, velocities, $\mathbb{P}_i$ and $\mathbb{G}$
3: Set the iteration number and parameters of the update function
4: **for** each iteration **do**
5: **for** each particle $\mathbb{X}_i$ **do**
6: **for** each $x_{id} \in \mathbb{X}_i$ **do**
7: Update the velocity using Eq. (24)
8: Update the position using Eq. (22) and (23)
9: **end for**
10: Calculate the fitness function $f()$
11: **if** $f(\mathbb{X}_i) < f(\mathbb{P}_i)$ **then**
12: $\mathbb{P}_i = \mathbb{X}_i$
13: **end if**
14: **if** $f(\mathbb{P}_i) < f(\mathbb{G})$ **then**
15: $\mathbb{G} = \mathbb{P}_i$
16: **end if**
17: **end for**
18: **end for**
19: Select the final $\mathbb{G}$ as the task allocation solution

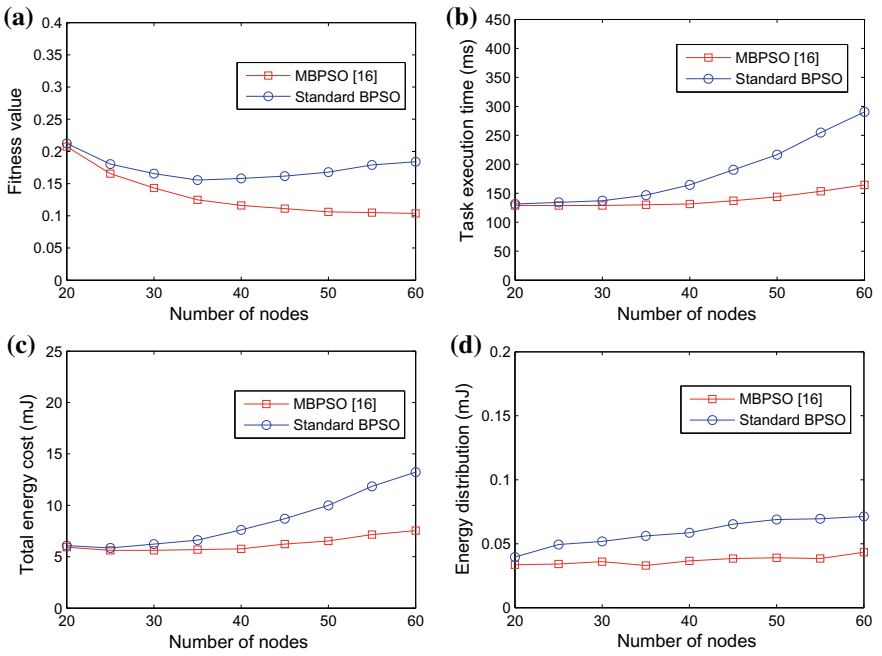


Fig. 18 Performance comparison between standard BPSO and MBPSO [16] (Recreated from [16].)

switching between 0 and 1 as expressed in Eq. (25).

$$x_{id}^{j} = \begin{cases} 1 - x_{id}^{j}, & \text{if } rand() < \phi \\ x_{id}^{j}, & \text{otherwise} \end{cases} \tag{25}$$

The performance comparison of adding the mutation operation and the standard BPSO is demonstrated in Fig. 18.

In contrast to the traditional heuristic task allocation algorithms as presented in Sect. 6.1, the bio-inspired heuristic algorithms perform better on the network lifetime improvement and energy balance among the sensor nodes. However, neither the GA or PSO guarantees the global optimum task allocation solutions. In addition, they need a large quantity of iterations of the generation to converge, which could consume longer time.

7 Conclusion

Nowadays, energy-aware task allocation methods are essential to improve the energy efficiency of complex WSN applications. The appropriate task allocation can efficiently balance the energy consumption of the networks, especially for the WSNs with complex computation tasks. In order to evaluate the task allocation approaches, the important metrics, e.g., network lifetime, residual energy distribution of nodes, time requirement and fault tolerance, need to be considered. The models of the WSN application tasks also play a fundamental role in the task allocation problem. Among the summarized task models, the most frequently used is the directed acyclic graph (DAG) model. Based on the DAG model, a common energy and time cost functions for the node executing the processes the tasks and the corresponding communications can be formulated. The presented taxonomy of the task allocation algorithms classifies the existing algorithms into two groups: the heuristic and optimal task allocation. Based on the evaluation metrics, the advantage and disadvantages of the presented approaches in each group are analyzed and compared.

In cluster-based WSNs, the optimal task allocation approaches are good choices in spite of being slightly more complex than the heuristic ones, since each cluster can be optimized independently and the cluster size is relatively small. In multi-hop mesh WSNs, heuristic algorithms are preferred. Although the traditional heuristic approaches are relatively simple, they are usually suffering from getting stuck in the local optimum. The bio-inspired heuristic algorithms like GA and PSO could provide high probability to obtain the global optimal solution. Nevertheless, they cannot guarantee the global optimum and need a large quantity of iterations of the generation to obtain the final solutions. Therefore, the task allocation approaches in WSNs still need to be improved to better match the requirements of the applications.

References

1. Raghunathan, V., Schurgers, C., Park, S., Srivastava, M.: Energy-aware wireless microsensor networks. IEEE Signal Process. Mag. **19**(2), 40–50 (2002)
2. Barr, K.C., Asanovi, K.: Energy-aware lossless data compression. ACM Trans. Comput. Syst. **24**(3), 250–291 (2006)
3. Tyagi, S., Kumar, N.: A systematic review on clustering and routing techniques based upon LEACH protocol for wireless sensor networks. J. Netw. Comput. Appl. **36**(2), 623–645 (2013)
4. Ren, F., Zhang, J., He, T., Lin, C., Ren, S.K.D.: EBRP: energy-balanced routing protocol for data gathering in wireless sensor networks. IEEE Trans. Parallel Distrib. Syst. **22**(12), 2108–2125 (2011)
5. Zhao, M., Yang, Y., Wang, C.: Mobile data gathering with load balanced clustering and dual data uploading in wireless sensor networks. IEEE Trans. Mobile Comput. **14**(4), 770–785 (2015)
6. Liang, Y., Peng, W.: Minimizing energy consumptions in wireless sensor networks via two-modal transmission. SIGCOMM Comput. Commun. Rev. **40**(1), 12–18 (2010)
7. Vecchio, M., Giaffreda, R., Marcelloni, F.: Adaptive lossless entropy compressors for tiny iot devices. IEEE Trans. Wirel. Commun. **13**(2), 1088–1100 (2014)
8. Yu, W., Huang, Y., Garcia-Ortiz, A.: An altruistic compression-scheduling scheme for cluster-based wireless sensor networks. In: 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), pp. 73–81, June 2015
9. Liu, C., Wu, K., Pei, J.: An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. IEEE Trans. Parallel Distrib. Syst. **18**(7), 1010–1023 (2007)
10. Huang, Y., Yu, W., Garcia-Ortiz, A.: PKF: a communication cost reduction schema based on Kalman filter and data prediction for wireless sensor networks. In: Proceedings of 26th IEEE International System-on-Chip Conference, CAS, pp. 73–78 (2013)
11. Huang, Y., Yu, W., Osewold, C., Garcia-Ortiz, A.: Analysis of PKF: a communication cost reduction scheme for wireless sensor networks. IEEE Trans. Wirel. Commun. **15**(2), 843–856 (2016)
12. Tang, L., Sun, Y., Gurewitz, O., Johnson, D.B.: PW-MAC: an energy-efficient predictive-wakeup MAC protocol for wireless sensor networks. In: INFOCOM, 2011 Proceedings IEEE, pp. 1305–1313, April 2011
13. Wu, Y., Li, X.Y., Li, Y., Lou, W.: Energy-efficient wake-up scheduling for data collection and aggregation. IEEE Trans. Parallel Distrib. Syst. **21**(2), 275–287 (2010)
14. de Meulenaer, G., Gosset, F., Standaert, F.X., Pereira, O.: On the energy cost of communication and cryptography in wireless sensor networks. In: 2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, pp. 580–585, October 2008
15. Shen, C., Plishker, W.L., Ko, D., Bhattacharyya, S.S., Goldsman, N.: Energy-driven distribution of signal processing applications across wireless sensor networks. ACM Trans. Sens. Netw. **6**, Article. 8, 32 (2010)
16. Yang, J., Zhang, H., Ling, Y., Pan, C., Sun, W.: Task allocation for wireless sensor network using modified binary particle swarm optimization. IEEE Sens. J. **14**(3), 882–892 (2014)
17. Guo, W., Li, J., Chen, G., Niu, Y., Chen, C.: A PSO-optimized real-time fault-tolerant task allocation algorithm in wireless sensor networks. IEEE Trans. Parallel Distrib. Syst. **26**(12), 3236–3249 (2015)
18. Shi, H., Wang, W., Kwok, N.: Energy dependent divisible load theory for wireless sensor network workload allocation. Math. Probl. Eng. **2012**, Article ID 235289, 16 (2012). https://doi.org/10.1155/2012/235289
19. Li, X., Liu, X., Kang, H.: Sensing workload scheduling in sensor networks using divisible load theory. In: IEEE GLOBECOM 2007—IEEE Global Telecommunications Conference, Washington, DC, pp. 785–789 (2007). https://doi.org/10.1109/GLOCOM.2007.152

20. Edalat, N., Tham, C.K., Xiao, W.: An auction-based strategy for distributed task allocation in wireless sensor networks. Comput. Commun. **35**(8), 916–928 (2012)
21. Edalat, N., Xiao, W., Tham, C.K., Keikha, E., Ong, L.-L.: A price-based adaptive task allocation for wireless sensor network. In: IEEE 6th International Conference on Mobile Adhoc and Sensor Systems, Macau, pp. 888–893 (2009)
22. Dai, L., Chang, Y., Shen, Z.: An optimal task scheduling algorithm in wireless sensor networks. Int. J. Comput. Commun. Control **6**(1), 101–112 (2011)
23. Huang, Y., Yu, W., Garcia-Ortiz, A.: Accurate energy-aware workload distribution for wireless sensor networks using a detailed communication energy cost model. J. Low Power Electron. **10**(2), 183–193 (2014)
24. Jin, Y., Jin, J., Gluhak, A., Moessner, K., Palaniswami, M.: An intelligent task allocation scheme for multihop wireless networks. IEEE Tran. Parallel Distrib. Syst. **23**(3), 444–451 (2012)
25. Yu, W., Huang, Y., Garcia-Ortiz, A.: Modeling optimal dynamic scheduling for energy-aware workload distribution in wireless sensor networks. In: Proceedings of 12th International Conference on Distributed Computing in Sensor Systems (DCOSS 2016), Washington DC, pp. 116–118 (2016)
26. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: 33rd Annual Hawaii International Conference on System Sciences, pp. 10. IEEE Press (2000)
27. Rappaport, T.: Ad Hoc Mobile Wireless Sensor Networks: Protocols and Systemes. Prentice Hall, Englewood Cliffs, NJ (1996)
28. Dai, L., Shen, Z., Chen, T. and Chang, Y.: Analysis and modeling of task scheduling in wireless sensor network based on divisible load theory. Int. J. Commun. Syst. (Washington, DC) **27**(5), 721–731 (2014)
29. Yu, W., Huang, Y., Garcia-Ortiz, A.: Distributed optimal on-line task allocation algorithm for wireless sensor networks. IEEE Sens. J. **18**(1), 446–458 (2018)
30. Jin, Y., Vural, S., Gluhak, A., Moessner, K.: Dynamic task allocation in multi-hop multimedia wireless sensor networks with low mobility. Sensors **13**(10), 13998–14028 (2013)
31. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Syst. Tech. J. **49**, Article 2, 291–307 (1970)
32. Fiduccia, C.M., Mattheyses, R.M.: A linear-time heuristic for improving network partitions. In: 19th Conference on Design Automation, pp. 175–181. IEEE (1982)
33. Hagras, T., Janecek, J.: A high performance, low complexity algorithm for compile-time job scheduling in homogeneous computing environments. In: 2003 International Conference on Parallel Processing Workshops, pp. 149–155 (2003)
34. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, vol. 5, pp. 4104–4108, Orlando, FL (1997)
35. Liu, J., Mei, Y., Li, X.: An analysis of the inertia weight parameter for binary particle swarm optimization. IEEE Trans. Evolut. Comput. **20**(5), 666–681 (2016)
36. Melanie, M.: An Introduction to Genetic Algorithms, Fifth printing. Massachusetts London, England, Cambridge (1999)
37. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co. (1989)
38. Tsiatsis, V., Kumar, R., Srivastava, M.B.: Computation hierarchy for in-network processing. Mobile Netw. Appl. **10**(4), 505–518 (2005)

Sensor Assignment to Missions: A Natural Language Knowledge-Based Approach

Alun Preece

Abstract A key problem in managing intelligence, surveillance and reconnaissance (ISR) operations in a coalition context is assigning available sensing assets—of which there are increasingly many—to mission tasks. High demands for information and relative scarcity of available assets implies that assignments must be made taking into account all possible ways of achieving an ISR task by different kinds of sensing. Moreover, the dynamic nature of most ISR situations means that asset assignment must be done in a highly agile manner. The problem is exacerbated in a coalition context because it is harder for users to have an overview of all suitable assets across multiple coalition partners. In this chapter, we describe a knowledge-driven approach to ISR asset assignment using ontologies, allocation algorithms, and a service-oriented architecture and we analyse the approach of using a representation based on Controlled English (CE) to improve the interface and human-in-the-loop aspects of the sensor assignment. An illustration of the use of the system from a mobile device is presented.

1 Introduction

We live in an age where unprecedented amounts of data are available to inform human decision-making. In the UK, *big data* was identified as the first of "eight great technologies" for economic growth.[2] In the USA, the Department of Defence listed its first science and technology priority for 2013–17 as *data to decisions* (D2D): "Science and applications to reduce the cycle time and manpower requirements for analysis and use of large data sets" [5]. The wording here emphasises that the data landscape changes rapidly and, to be effective, "big data" techniques—including data collection, analytics and visualisation—need to be highly agile. The typical model

[1] https://www.gov.uk/government/publications/eight-great-technologies-infographics.

A. Preece (✉)
Crime and Security Research Institute, Cardiff University, Cardiff, UK
e-mail: PreeceAD@cardiff.ac.uk

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_7

Fig. 1 An abstract data-to-decision pipeline

for a D2D pipeline is shown in Fig. 1 where data are collected from one or more data sources of various kinds, processed by a variety of analysis services and the results delivered to the decision-maker in some actionable form.

Available data sources—often characterised by the "three Vs", volume, velocity and variety [14]—span an enormous range of types, including physical sensors, geospatial and other information systems, social media of many kinds and human sources. Often it is necessary to combine data from multiple heterogeneous sources, through some process of information fusion [16]. Analytic services are equally diverse, including signal processing, statistical, machine learning and inferential systems. Again, often multiple analytic processes are used in combination, for example signal processing to identify lower-level features, followed by inference to perform higher-level classification. The optimal form for information retrieval and delivery to a human decision-maker depends on both human capabilities and system capabilities. Contrary to intuition, providing more information does not necessarily improve human decision-making [1, 9, 12]. Thus, a summary-level representation of information, with the ability to drill-down to see rationale and supporting evidence, is key to supporting effective human decision-making. The physical hardware for accessing the system is another consideration for the form of information: for example, delivery to a mobile user via a smartphone app or wearable device requires different human–computer interaction approaches than delivery to a large, conventional display screen.

In addition to work on techniques for data collection, processing and dissemination, there has also been significant investment in tools and methods to make it easier and quicker for developers to construct D2D pipelines, including research in middleware, platforms and automated workflows [6]. However, the majority of work in this space has taken a data-driven view. A problem that has received less attention is how to rapidly construct pipelines by working backwards from an intended decision (or hypothesis or query) and identifying useful analysis services and underlying data sources that can assist the decision-maker [8, 24]. The collection and availability of information are necessary, but not sufficient for assisting the decision-maker. For optimal decision-making, the (human) search costs must be minimised; that is, the decision-maker must be able to access information in a timely manner [7].

This kind of rapid engineering of data analytics pipelines from sensing sources in response to urgent information needs can be seen in well-publicised incidents such as the damage to Japan's Fukushima nuclear plant in the wake of the 2011 earthquake. An urgent need arose to monitor radiation leaks—the decision-maker's intent—

Fig. 2 Constructing a D2D pipeline dynamically by a "backward chaining" process

leading to the rapid deployment of networked Geiger counters, many of which were private devices shared via Internet of things approaches.[3] This "backward chain" from decision intent to data sources is shown in Fig. 2. Note that the arrows here depict control flow: the user's intent frames the problem, leading to the selection of suitable services and compatible data sources. The result is a dynamically constructed pipeline as shown in Fig. 1. Building this pipeline on-the-fly through a highly automated process of service and source selection, and composition is a method to address the original priority to "reduce the cycle time and manpower requirements" in D2D.

A number of trends seem to point towards an even more flexible and agile view of D2D systems. Firstly, the data sources are becoming increasingly "smart" and communicative. Autonomous vehicles and robotic systems, together with increasingly computationally capable Internet of things devices operating in peer-to-peer networks open up greater potential for collective intelligence and self-organisation at what has traditionally been the edge of the network, where data sources are often located. At the same time, the rapidly increasing sophistication of mobile devices has freed decision-makers to operate in contexts much nearer the tactical edge. Mobile users have become adept at agile, on-the-fly decision-making, able to cope with dynamically changing sets of requirements while simultaneously carrying out actions in the field. Many activities previously seen as strategic or operational in decision-making terms, which have been "tacticalised" by mobile technology. Pervasive information sources have given rise to a new generation of context-aware, assistive technologies typified by Apple's Siri[4] and Google Now.[5] These technologies are changing the modes of interaction between users and their devices, with the device able to take an increasingly active role in the interaction, for example, by engaging in conversation or pushing notifications to the user in an anticipatory manner.

In this context, the traditional D2D pipeline can be re-thought as a collection of interactions between agents with different specialisms: the data sources, analytic services and decision-makers can be viewed as engaging in peer-to-peer interactions with each other, with chains of interaction able to start anywhere in the network and flow in any direction, from data to decision, or from query to response. This "conversational D2D" model is shown in Fig. 3. The different styles of "speech

[3] http://www.wired.com/opinion/2012/12/20-12-st_thompson/.

[4] https://www.apple.com/ios/siri/.

[5] http://www.google.com/now/.

Fig. 3 Conversational D2D

bubble" here suggest that the machine participants will tend to communicate in structured message formats, while the human participants will use a more natural form of interaction.

In the context of intelligence, surveillance and reconnaissance (ISR) operations, there are typically multiple ways to achieve a task using sensor-provided data. For example, the National Imagery Interpretability Rating Scales (NIIRS) framework characterises various kinds of ISR tasks that can be achieved using visual sensing data of different types (visible, radar, infrared and multispectral) [13]. Other kinds of sensor-provided data can be similarly characterised, for example acoustic and seismic data. Therefore, given an ISR task and a set of sensing assets in a particular area of interest, there may be many options for resourcing that task. In a coalition context, the problem is more complex, because the assets may be "owned" by different partners. From the point of view of a user (e.g. an ISR analyst) with a particular information need (e.g. tracking high-value targets in an area), the problem of identifying suitable ISR assets is difficult, without a great deal of knowledge about sensing capabilities and availability of coalition assets.

As part of a solution to this problem, several works have proposed the use of some form of knowledge base or mapping that relates sensor capabilities to task requirements (e.g. [15, 20, 28, 31]), to aid either automatic or semi-automatic identification of suitable assets for tasks. In this chapter, we present an approach founded on the Military Missions and Means Framework (MMF) [29] using ontologies of task and asset types, and an automatic procedure for matching one to the other, through the capabilities required on one and provided by the other. This approach was intended to be extensible, and we have shown that additional matching knowledge from the NIIRS framework can easily incorporated [18].

Our approach—called sensor assignment to missions (SAM)—is intended mainly for automated assignment of assets to tasks, in a context where ISR assets are relatively scarce (demand exceeds supply), and many tasks may be competing for the same resource. While assets may be shared among tasks, we initially assumed that resourced tasks would not be shared among users (i.e. while serving a task, an asset is exclusively assigned to a single user). Because there was competition between tasks, we studied models for the pre-emption of existing tasks by new tasks [21]. A typical case would involve a new higher-priority task effectively "stealing" resources from an existing lower-priority task.

Going forward, in line with the above ideas of "conversational D2D", we relaxed some of those assumptions to support a more interactive SAM approach, where the human user works in a more cooperative manner with the system. Specifically, we wanted users to be able to explore the various means of achieving a task and consider the option of sharing existing (identical or similar) tasks rather than creating a new request for resources (thus avoiding common cases of pre-emption). The main idea is to show users—subject to access policies—currently resourced tasks in their area of interest and to make it easier for them to "join" an existing task (again, subject to access policies) than to create a new one. For this to work, the representation of tasks and means of achieving them (combinations of assets) would need to be much more transparent to users than in our earlier work and other work in this area.

To support this more human-in-the-loop approach, we experimented with the use of a *controlled natural language* (CNL) to express the elements of our knowledge base and allow us to generate human-understandable representations of ISR tasks and their resourcing. A CNL is a subset of a natural language, commonly English, with restricted syntax and vocabulary. Often they are used to provide an information representation that is easily machine processable (with low complexity and no ambiguity) while also being human-readable (see, e.g., [32]). Our main goals were:

- to verify whether our MMF-based knowledge base could be expressed in CNL, with no loss of power to support automated asset-task matching and
- to explore how a CNL-based representation of tasks and their resourcing could be used to create a human-understandable tool to promote task sharing among users.

The chapter is organised as follows: Sect. 2 reviews our MMF-based approach to task-asset matching, including the task and asset ontologies and the NIIRS-based matching procedure; Sect. 3 provides a full exposition of the matching algorithm as a logic program in Prolog; Sect. 4 describes how we achieved the goal of re-expressing our knowledge base and the various facts associated with task-asset assignment in a CNL; Sect. 5 presents our prototype tool—in the form of a mobile tablet-based app—that allows users to explore the space of assigned tasks and choose to share existing tasks; finally, Sect. 6 concludes the chapter.

2 Knowledge-Based Matching of ISR Assets to Tasks

Our original knowledge base and task-asset matching procedure [10, 23] was based on an ontology derived from the Military Missions and Means Framework (MMF) [29]. To derive our ontology, we formalised concepts and relationships from the MMF documentation, the main ones being as shown in Fig. 4. Missions are comprised of operations which are in turn comprised of tasks. Tasks require capabilities, which are provided by assets. Assets include platforms and systems; systems—including sensors—are comprised of components and mounted on platforms. The original ontology included a relationship "allocated to" to capture that an asset was assigned to resource a particular task. The ontology was implemented in OWL DL

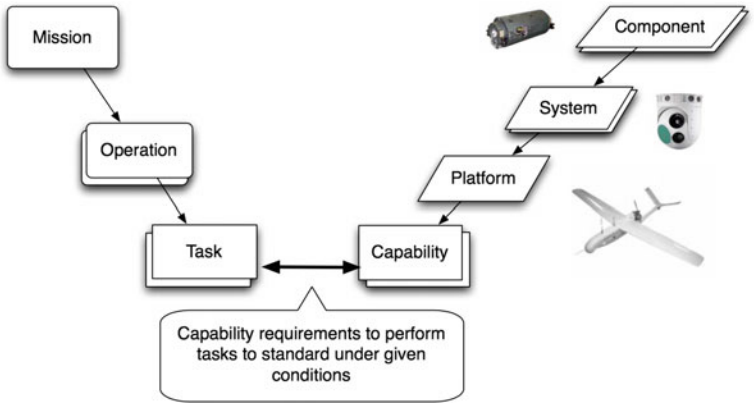


Fig. 4 Military missions and means framework

version 1[6], and the task-asset matching procedure was implemented using a combination of Pellet[7] and Java. Pellet was used to perform classification on the ontology and to infer all capabilities provided by all assets (sensor and platforms). The remainder of the matching procedure was implemented in Java, based on a set-covering algorithm [10].

Although it is possible to imagine a single all-encompassing "ISR ontology", we adhered to the Semantic Web vision of multiple interlinking ontologies covering different aspects of the domain: sensors, platforms, tasks, etc. This allowed us to build on a substantial body of pre-existing work but, as we will see, left us with the problem of relating sensors/platforms to tasks. To fill this gap, we provided an ontology based on the Military Missions and Means Framework (MMF) [29], which is essentially a collection of concepts and properties to reason about the capabilities required to accomplish a mission (e.g. mission, task, capability, asset, etc.). MMF was developed by the US Army Research Laboratory to provide a model for explicitly specifying a military mission and quantitatively evaluating the utility of alternative means to accomplish it. Ours were the first attempt to define an ontology based on the framework, using MMF allowed us to benefit from its familiarity to users. The way MMF describes the linking between missions and means—shown in Fig. 5—naturally fitted the notion of matchmaking: on the one hand, we have missions breaking down into operations, and operations into tasks, where each task may require different capabilities to be accomplished; on the other hand, we have the capabilities provided by assets as a result of aggregating the capabilities of its constituent systems and subsystems.

Figure 5 illustrates the main concepts of our MMF ontology. On the left-hand side, we have the concepts related to the mission: a mission comprises several operations to be carried out, and each operation breaks down into a number of tasks that must be

[6] http://www.w3.org/TR/owl-guide/.

[7] An OWL description logic reasoner, http://clarkparsia.com/pellet/.

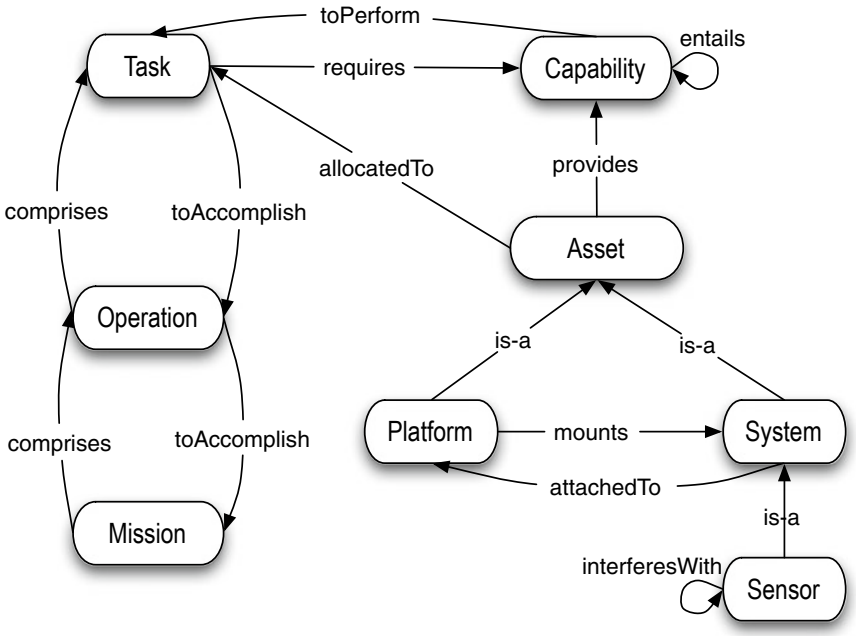


Fig. 5 Missions and means framework ontology [10, 23]

accomplished. On the right-hand side, we have concepts related to means: a sensor is a system that can be attached to a platform; inversely, a platform can mount one or more systems; both platforms and systems are assets; assets provide capabilities; a capability can entail a number of more elementary capabilities and is required to perform certain type of tasks; and inversely, a task is enabled by a number of capabilities; some sensors can interfere with other sensors, so they cannot be used simultaneously; and finally, at some point, assets will be allocated to specific tasks that require the capabilities provided by them.

Note that all concepts shown in Fig. 5 are general MMF concepts with the exception of sensor, which we introduced (as a refinement of the MMF core concept system) in order to link the MMF ontology with the ISR domain-specific ontologies. This is because, while the MMF ontology describes the main concepts used in our matchmaking framework and is generic to military and military-style missions and means in the widest sense, in order to describe specific instances of those concepts we needed domain-specific vocabulary discussed in the next section.

There was already a sizeable amount of work done in providing descriptive schemas and ontologies for sensors, sensor platforms, and their properties, such as SensorML [3], OntoSensor [28], CIMA [17], and the MMI Platforms ontology [2] among others. There were also several well-known structured descriptions of tasks in the military missions context, most notably the US Universal Joint Task List (UJTL), the CALL thesaurus, the US JC3IEDM model and the UK JETL/METL task lists.

These existing representations provided partial coverage of what we needed to model but, as they originated in either the sensor/platform or task spaces, they lacked knowledge of how capabilities provided by the types of sensors/platforms may satisfy the capabilities required by tasks. Consequently, our approach was to reuse existing concept sets and to extend these with representations of capabilities.

During the knowledge analysis and acquisition stage, we found a number of issues that we have taken into account in our approach to the representation of the ISR domain, including the following:

- The absence of standardised taxonomies, and the existence of alternative, sometimes inconsistent, classifications for the same concepts.
- Existing attempts to conceptualise the domain were based on different dimensions, and more usually, several dimensions were mixed. For example, unmanned air vehicle (UAV) classifications tended to mix dimensions such as size or weight (e.g. MicroUAV vs. MiniUAV), performance (e.g. medium altitude vs. high altitude), task type (maritime reconnaissance, wide area surveillance, etc.) or ad hoc features such as their landing and take-off capabilities.
- There were fuzzy concepts that were difficult to classify as a single category. For example, light detection and ranging (LIDAR) is a type of sensor that has properties of both optical sensors and radars.
- Concepts that were supposed to refer to the same aspect of the domain were described at different abstraction levels. Closely related to this issue was the tension between considering a concept as primitive, or as a composition of more basic elements; for example, a reconnaissance capability might be seen as implying a combination of mobility and sensing capabilities.

Figure 6 shows an overview of the SAM approach. The process operates clockwise from the top-left of the figure. The system (or user) specifies a task to localise SUVs in a particular area. Internally the matching procedure uses the models and knowledge base to determine those assets with a *Visible NIIRS rating of 4* (or higher) or an *Acoustic NIIRS rating of 6* (or higher) can achieve this task. These ratings are provided by: a *UAV platform* with *video camera sensor*, or an *unattended ground system platform* with an *acoustic array sensor*. In some cases, a collection of asset types are needed to fully meet the requirements of tasks; we refer to such collections as bundle types.

Bundle instances are now identified and ranked with a utility function for this specific task. A bundle instance may contain more than one instantiation of the bundle type, where more than one set of deployed assets is needed to achieve the task. Based on the ranking process, the results are presented back to the user for their confirmation. This aspect of the process could be automated too, choosing the highest utility solution, or the first above a certain threshold, but for this demonstration, we wanted to highlight the power of semi-automatic allocation with the human and machine agents working together as part of a hybrid team.

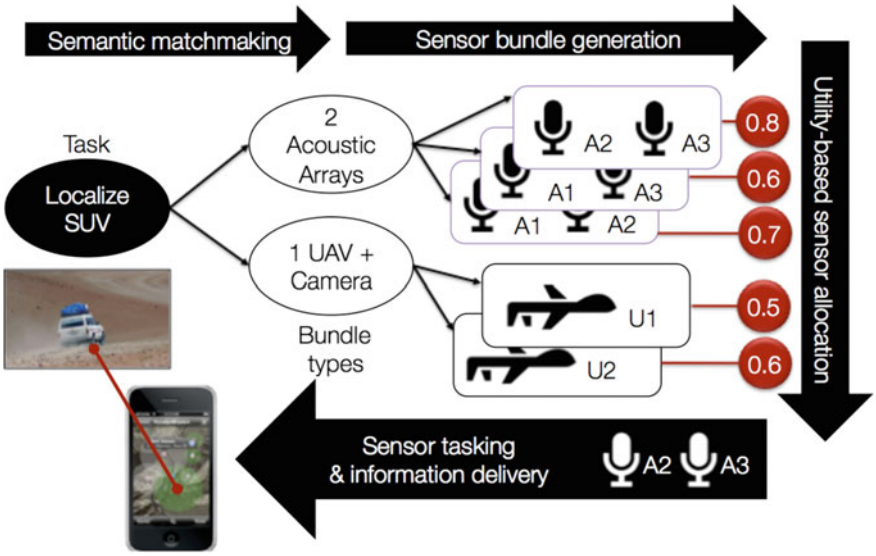


Fig. 6 Overview of the sensor assignment to missions (SAM) approach

2.1 NIIRS-Based Task-Asset Matching Knowledge

A revised matching procedure based on the NIIRS framework was introduced in [18]. Here we describe a revised implementation of the procedure in Prolog, for use with the CNL knowledge base introduced in the following section. The knowledge base contains a set of *intelligence clause* tuples of the form: $\langle IC, DS, FS, C, IT, NR \rangle$, where:

- IC is an *intelligence capability* which include the three capabilities in NIIRS—*detect*, *distinguish*, *identify*—and also *localize*;
- DS is a set of *detectable things* drawn from the NIIRS framework (e.g. kinds of vehicle or building);
- FS is a set of more specific features of the detectable entities (e.g. the roads or guard posts of a base, or the runways of an airport);
- C is a context, defining the preconditions that must hold for the intelligence clause to apply (e.g. detection of a ship in the context of open water);
- IT is a type of *sensor-provided data* which includes the NIIRS types—*visible*, *radar*, *infrared*, *multispectral*—plus other types including *acoustic* and *seismic*; and
- NR is a NIIRS rating on a scale of 0–9 (e.g. *visible-6* or *radar-4*).

These intelligence clause tuples are derived from the NIIRS framework, augmented with others drawn from the sensing literature (e.g. on acoustic sensing).

2.2 Matching Procedure

We define a *task type* as a pair $\langle IC, DS \rangle$ where IC and DS are defined as above. Some example task types are *detect* {*tank*} and *distinguish* {*tank, jeep*}. Note that task types feature either a single detectable (for *detect*, *identify* and *localize* tasks) or a pair of detectables (for *distinguish* tasks).

We say that a given task type $TT_i = \langle IC_i, DS_i \rangle$ *requires* a set of capabilities $CT_i = \{IT_j, NR_j\}$ if there is an intelligence clause fact $IC_j = \langle IC_j, DS_j, FS_j, C_j, IT_j, NR_j \rangle$ where $IC_i = IC_j$ and $DS_i \subseteq DS_j$.[8]

We define a *bundle type* as a pair $\langle PT, SS \rangle$ where PT is a platform type and SS is a set of sensor types (according to the ontology in Fig. 5), and the ontology contains the statement PT *mounts* ST_j for each $ST_j \in SS$—that is, each sensor type in the set can be mounted on that platform type.[9]

We say that a given bundle type $BT_k = \langle PT_k, SS_k \rangle$ *provides* a set of capabilities CB_k if, for each $C_l \in CB_k$, our ontology includes either the statement PT_k *provides* C_l or the statement ST_j *provides* C_l for some $ST_j \in SS_k$. In other words, each capability in CB_k is provided either by the platform type or a sensor type in the corresponding bundle type BT_k.

We say that a bundle type BT_k *matches* a task type TT_i if $CB_k \supseteq CT_j$ according to the above definitions—that is, if the set of capabilities provided by the bundle type contains the set of capabilities required by the task type. We say that a bundle type BT_k *minimally matches* a task type TT_i when no sensor type can be removed from the bundle type such that the *matches* relationship still holds.

Note that FS and C, while part of the NIIRS-based model for completeness, are not part of the implemented matching procedure defined here. In principle, it would be trivial to include them as additional matching constraints; in practice, FS is unused because users to express tasks in terms of detecting entire detectable entities (rather than specific features thereof) and C is redundant because the context (e.g. river vs. open water) is determined by the area of interest—see Sect. 3.

2.3 KB Table Generation and Asset Assignment

The types of task and asset are relatively static; for this reason is it reasonable to pre-generate a *lookup table* of all possible task type/bundle type minimal matches. Doing so allows us to deploy the knowledge base on a mobile device as described in [21], where we compute the size of a realistic complete table to be in the order of

[8] In some cases, a hierarchy of detectables allows some inference here; for example, any clause involving *detect* and a kind of detectable D is considered to cover all more specialised kinds of D also—so *detect* { *car* } covers specialised kinds of car (*jeep, SUV, saloon*, etc.).

[9] The ontology contains the additional relationship *interferes with* to cover cases where types of sensor are incompatible. No valid bundle type can contain pairs of sensor types involved in an *interferes with* relationship.

20,000 entries, which can comfortably be stored in 12 MB on a smartphone or tablet. Note that the size of the deployed table can easily be reduced by including only task types and asset types relevant to a particular mission context. It would not make sense, for example, to include maritime ISR tasks and assets in a land-based context. We refer to the individual entries in the lookup table as *assignment templates*. These are of the form: $\langle IC, DS, BT, UF \rangle$, where:

- IC is an intelligence capability, as above;
- DS is a set of detectable things, as above;
- BT is a bundle type, as above, such that BT minimally matches the task type formed by $\langle IC, DS \rangle$;
- UF is a *utility function* compatible with BT and the task type formed by $\langle IC, DS \rangle$.

The utility function associated with an entry provides a means of assessing how effective a particular instance of the bundle type is likely to be in achieving a particular instance of the task type. Usage of the lookup table in asset assignment is covered in detail in [21]. In outline, the assignment procedure is as follows:

1. A user creates a task T_i, from which the system derives the corresponding task type $TT_i = \langle IC_i, DS_i \rangle$.
2. The system retrieves all entries $\langle IC_j, DS_j, BT_j, UF_j \rangle$ where $IC_i = IC_j$ and $DS_i \subseteq DS_j$.
3. The system determines all possible *bundle instances* that conform to all retrieved bundle types BT_j and uses the corresponding utility functions UF_j to derive a utility for each.
4. The preceding step is performed as part of a distributed allocation protocol based on [30] that attempts to maximise overall utility in the face of multiple competing tasks. As explained in Sect. 1, this may involve pre-empting one or more existing tasks to accommodate the new task.

Having reviewed our task-asset matching and assignment procedures at a high level, we will now look at the matching algorithm in detail, by walking through its specification and implementation as a Prolog logic program.

3 The SAM Matching Algorithm

This section describes the specification and implementation of the sensor assignment to missions (SAM) matching procedure. The remainder of the chapter is understandable without the details of this section, so readers may choose to skip to Sect. 4. This implementation includes the generation of the KB lookup table for the mobile version of SAM shown in Sect. 5. This version of SAM was a development of the NIIRS-based matcher first published in [18]. The main features of this implementation are:

1. Tasks are represented as in [25], following the model of NIIRS interpretation tasks. Each task has four elements, which we capture using properties:

- `hasIntCapability` refers to the intelligence capability, which includes the three NIIRS types[10]—`detect`, `identify`, `distinguish`—and the `localize` type used for 2D-localisation tasks [27];
- `hasDetectables` refers to the set of target objects, which must contain at least two types of object for `distinguish` tasks and at least one type of object for the other three kinds of int. capability (though these cardinality constraints are not expressed in the Prolog model);
- `hasAreaOfInterest` refers to the area of interest for the task;
- `hasTimePeriod` refers to the required timing for the task.

The latter two properties are not used in the matching procedure, as they are relevant only to assigning asset instances. They are included for completeness with respect to [25]. Two sample tasks using these properties are provided in the "test data" section of the code.

2. The ontology includes the set of `detectable` things as described in [25]. All platforms—including sensor platforms themselves—are subclasses of `detectable`. (Sensors are considered too small to be detectable.) While there are other kinds of "detectables" in the NIIRS documentation, we focus on vehicles.
3. The "mounts" property from the 2008 published version of SAM [10], which relates platforms to sensors, has been renamed to `carries` in line with later implementations.
4. Tasks are now matched to assets by a two-stage process: (1) a NIIRS-style knowledge base (the set of `intClauses`, similar to [25]) gives the intelligence type (`visible`, `radar`, `acoustic`, etc.) and NIIRS-style rating (0–9) required for the task; (2) these capabilities are matched to sensor and platform types using the `providesCapability` clauses. The knowledge base of `intClauses` is not intended to be complete, but is derived from the set of clauses in the original rule-based implementation of the NIIRS matching procedure [18].
5. In the NIIRS approach, higher-rated capabilities entail lower-rated ones for the same type (e.g. `radar-6` entails `radar-5` and lower).
6. As described in [25], NIIRS-style clauses can carry information on specific features of an object that can be determined from intelligence of the given type, and also the context in which the interpretation of the data can occur. We ignore these in this implementation (the former are irrelevant as we are interested only in targeting the detectable objects, not specific features of them; the latter could be used in more complex assignment situations involving use of GIS data as background information, or sensor cueing, but are outside the scope of this implementation).
7. Pairs of intelligence type and rating are always matched to platform configurations (a single platform with one or more sensors, known as a *bundle type* in our later work), so there is no longer any need for the notion of "package configuration" from the 2008 version of SAM.

[10] http://www.fas.org/irp/imint/niirs_c/guide.htm.

8. As described in [21], the combination of a task type and platform configuration (i.e. bundle type) determines the choice of joint utility model (JUM). Here we simplify this choice to three rules:
   - any localisation task can use the `twoDimensionalLocalization` JUM only;
   - any bundle type containing an acoustic sensor can use the `twoDimensionalLocalization` JUM;
   - any task other than localisation can use the `cumulativeDetectionProbability` JUM.

3.1 Sample Queries

- To generate the lookup table described in [21]:

```
generateKBTable.
```

  This provides a list of ⟨*task type*, *bundle type*, *JUM*⟩ triples.
- To obtain a single platform configuration that covers the aggregate capabilities for a task:

```
matches( Task, PlatformConfig ).
```

  Alternative package configurations will be generated by backtracking.

3.2 Code Walkthrough

See the appendix for the complete source code. Brief explanations of each predicate used in the matching procedure follow.

- `generateKBTable` cleans up any previously-asserted tasks and

```
kbTable( Task, BundleType, JointUtilityModel )
```

  tuples, then proceeds via backtracking to generate all possible tasks, find matches and JUMs for them and assert any it has not seen before; finally, it lists the `kbTable` triples.
- `generateIstarTask` constructs and asserts an `intTask` instance comprising an int. capability and set of detectables.
- `intTaskType` generates an int. capability and corresponding set of detectables (a pair of detectables for `distinguish` tasks; a single detectable for all other int. capabilities).

- `matches` aggregates the required capabilities for a task, generates a platform configuration (PC), tests if the PC provides the aggregate capabilities and tests that the PC is minimal in terms of covering the capabilities.
- `requiresCapabilities` aggregates the capabilities for the given task using the NIIRS-style `intClause` knowledge base: a clause is applicable if the set of detectables for the clause is a superset of those for the task.
- `platformConfiguration` generates a platform configuration by first choosing a platform type, then generating a list of sensor types that is a subset of all sensor types that can be carried by that platform, and testing that the subset is actually configurable (there are no conflicts) for that platform. Note that a platform configuration is a list where the head is the platform type and the remainder is a non-empty set of sensor types.
- `isConfigurable` fails if there is a subset of sensors that is not configurable for that platform.
- `unconfigurable` tests if there is an ontology `interferesWith` clause that indicates either: a pair of sensor types is always incompatible, or a pair of sensor types is incompatible for a specific platform type (e.g. because they cannot both be mounted at the same time).
- `providesCapabilities` succeeds if each given capability is provided by some asset in the given platform configuration.
- `capabilityProvidedByPlatformConfig` succeeds if the given capability is provided by some asset in the given platform configuration, using the ontology predicate `providesCapability`.
- `minimalConfiguration` succeeds if there is no proper subset of the sensors in the given platform configuration that provides the given set of capabilities.

The predicates `assertIfNew`, `writeKBTable` and `subseq0` are trivial "utility" predicates. The following clauses are part of the ontology and NIIRS-style knowledge base:

- `transitiveSubClassOf` defines the transitive subclass relationship.
- `carries` is a property that relates platform types to the sensor types that platform can carry. Any asset that can be carried by a parent platform can also be carried by its children.
- `subClassOf` defines direct parent–child class relationships; the definitions here include sensor, platform and detectable classes (where the latter include all platforms).
- `intCapability` defines the four basic ISTAR task capabilities.
- `intClause` is the NIIRS-style knowledge base. As described in [25], each clause has 6 elements:

  1. the int. capability;
  2. the set of detectables (exactly two for `distinguish` tasks, one for all other tasks);
  3. an optional list of more specific features of the detectables (not modelled here);
  4. an optional context for the interpretation task (modelled but ignored here);

5. the type of applicable intelligence data; and
6. the required rating for data of the given type.

Some of these clauses are derived from the published NIIRS tables, as described in [18]. The clauses involving acoustic sensing are not part of published NIIRS documentation, but are based on ITA literature (e.g. [11]). There are also two "general" clauses:

– if a class of detectable object can be detected by a given int. type and rating, then so can subclasses of that detectable;
– if an object of a class can be detected, we assume it can also be localised (given multiple sensor instances to perform, for example, 2D-localisation).

- `providesCapability` is a property that relates platform or sensor types to capabilities (int. types and ratings). Following [18], we associate int. types with kinds of sensor and ratings with platforms. These definitions include rules for capabilities that are entailed where one capability entails another, or where a child inherits its parent's capabilities.
- `entailsCapability` defines which capabilities are entailed by other capabilities; specifically, it handles entailment of ratings (e.g. `radar-5` entails `radar-4` and lower).
- `matchJointUtilityModel` defines how int. capabilities and bundle types (i.e platform configurations) relate to JUMs.

4 Reformulating the Knowledge Base in Controlled English

Several controlled natural languages exist; for our work, we selected a form of Controlled English known as ITA Controlled English, which arose from our research in the International Technology Alliance (ITA) [19]. ITA CE was chosen to aid knowledge reuse and the potential for system integration, because it was already in use in a number of related research projects. Hereafter, we refer to ITA CE simply as CE. The purpose of CE is that it provides a human-friendly information representation format that is directly processable by machine agents with a clear and unambiguous underlying semantics. In terms of machine processability, we aimed to demonstrate that CE can play the same role as OWL DL in formalising our ontology definitions,[11] while being more easily understandable by humans (in support of our second goal in Sect. 1).

Here we introduce CE in the context of reformulating the ontology definitions introduced in the previous section. All of the CE examples used in this chapter are directly processable by machine agents and, we hope to illustrate, more consumable by human readers than non-CNL equivalent technical representations. The

[11] Which in turn is similar to other efforts including SensorML [3], OntoSensor [28] and the W3C Semantic Sensor Network Incubator Group [15].

improvement of CE syntax to allow further linguistic variety and expressivity without undermining the unambiguous semantic grounding is a topic of current research.

4.1 A Brief Introduction to CE

CE is used to define both models and instances. Model definitions take the form of concept definitions. CE `conceptualise` sentences are intended to define by concepts by example; that is, they provide generalised examples of how to say things about concepts. Relationships between concepts are also considered to be concepts, though we will refer to them in this chapter simply as relationships. A CE model may also include the definition of logical inference rules (not shown in this chapter) which are used to express further information about the concepts and relationships and how they are logically related. Concepts may be specialisations of other concepts (indicated by `is a` declarations).

CE is approximately as expressive as the W3C's Web Ontology Language (OWL) [34]; its rule language has similar capabilities to the Semantic Web Rule Language [33].

For illustration, a sample CE model definition from the domain of social media analytics (adapted from [26]) is shown below.

```
conceptualise a ~ twitter account ~ A that
  is an online identity and
  is a temporal thing and
  has the value L as ~ location ~ and
  has the value NT as ~ number of tweets ~ and
  has the web image PP as ~ profile picture ~ and
  has the value NT as ~ number of tweets ~ and
  has the value NFR as ~ number of friends ~ and
  has the value NFO as ~ number of followers ~.
```

A `conceptualise` sentence defines a new concept in a CE model (ontology). New terms in the model—concepts, properties and relationships—are introduced between the tilde (~) symbols. The example defines the concept `twitter account` as being a child of the parent concepts `online identity` and `temporal thing` and having properties such as `location`, `number of tweets`, `profile picture`. The property definitions include the type of the value: either a literal value (e.g. for `number of tweets`) or a concept type (e.g. `web image` for the `profile picture` relationship).

Instances (facts) are defined using a similar syntax. The example below shows an instance of the concept `journalist`. (This example was chosen because the individual is a public figure and the BBC publicly lists the professional Twitter accounts of its journalists.)

```
there is a journalist named 'Jane Smith' that
  uses the twitter account 'jsmithbbc' and
  works for the media organization 'bbc'.
```

This instance is named `Jane Smith` and has a `uses` relationship with an instance of the concept `twitter account` (as defined above). The `twitter account` is named `jsmithbbc`. The `journalist` instance `Jane Smith` also has a `works for` relationship with an instance named `bbc` of the concept `media organization`. The role of CE is to have extensible models with whatever concepts, properties and relationships are needed. So `works for` is just one relationship that we chose to model, but there can be any of these.

4.2 Representing the Core MMF Ontology in CE

The following sample definitions cover part of the original MMF ontology (Fig. 5):

```
conceptualise a ~ capability ~ C.

conceptualise the mission M
 ~ comprises ~ the operation O.

conceptualise the operation O
 ~ comprises ~ the task T.

conceptualise the task T
 ~ requires ~ the capability C.

conceptualise the asset type A
 ~ is rated as ~ the NIIRS rating R and
 ~ provides ~ the capability C.

conceptualise a ~ system type ~ S that
 is an asset type.

conceptualise a ~ sensor type ~ S that
 is a system type.

conceptualise a ~ platform type ~ P that
 is an asset type.

conceptualise the platform type P
 ~ mounts ~ the system type S.
```

Here are some sample platform and sensor type definitions that extend the higher-level definitions; these are intended to be illustrative and are not complete. CE sentences beginning with "Note:" are annotations referring to terms in the preceding sentence.

```
conceptualise a ~ UAV ~ U that
 is a platform type.

conceptualise a ~ MALE UAV ~ M that is a UAV.
Note: MALE = Medium Altitude, Long Endurance.

conceptualise a ~ Predator A ~ P that
 is a MALE UAV.

conceptualise an ~ EO camera ~ E that
 is a sensor type.
Note: EO = Electro-optical.
```

Once the conceptual model is defined subsequent assertions can be made according to these concepts and relationships. The CE syntax `there is a…` is used to create instances. As an example, here is a "prototypical" instance `Predator A platform type` which defines all of the "prototypical" instances of sensor types that can be mounted on that kind of platform (specified using the `mounts` relationship):

```
there is a Predator A named
  'Predator A platform type' that
 mounts the sensor type
  'EO camera sensor type' and
 mounts the sensor type
  'TV camera sensor type' and
 mounts the sensor type
  'FLIR camera sensor type' and
 mounts the sensor type
  'LADAR sensor type'.
```

4.3 Representing Task-Asset Matching Knowledge in CE

To support the NIIRS-based matching of tasks to assets, we extend the above definition of task as follows, to include a NIIRS-style intelligence capability and one or more kinds of detectable thing. We also include a spatial area of interest, a time period, and a priority to allow tasks to be ranked if assets are scarce:

```
conceptualise the task T
~ requires ~ the intelligence capability IC and
~ is looking for ~ the detectable thing DT and
~ operates in ~ the spatial area SA and
~ operates during ~ the time period TP and
~ is ranked with ~ the task priority PR.
```

Here are some sample intelligence capabilities and detectable things:

```
there is an intelligence capability named
 detect.
there is an intelligence capability named
 identify.
there is an intelligence capability named
 localize.

there is a detectable thing named
 'wheeled vehicle'.
there is a detectable thing named
 'tracked vehicle'.
there is a detectable thing named
 'field artillery'.
```

Here is a sample task instance:

```
there is a task named t1265 that
 requires the intelligence capability detect
 and is looking for the detectable thing
  'wheeled vehicle' and
 operates in the spatial area r942 and
 operates during the time period t1789 and
 is ranked with the task priority medium.
```

The time period `t1789` and spatial region `r942` are assumed to be defined by separate instances containing the relevant spatio-temporal values, but these are not shown here.

As described in Sect. 2.1, the NIIRS-based approach allows automatic matching of tasks to asset capabilities, by means of encoding NIIRS knowledge as a set of *intelligence clause* facts. In CE, these facts are modelled as follows[12]:

```
conceptualise the intelligence clause IC
~ fulfills ~ the intelligence capability IC and
```

[12]Here we show only the 4 elements of the tuple $\langle IC, DS, FS, C, IT, NR \rangle$ from Sect. 2.1; FS and C are outside the scope of this chapter and omitted.

```
~ is looking for ~ the detectable thing DT1 and
~ provides ~ the capability C and
~ is rated as ~ the NIIRS rating R.
```

Here is an example *intelligence clause*, for the case that wheeled vehicles can be identified with visible imagery at NIIRS rating 4 or better:

```
there is an intelligence clause named ic03 that
 fulfills the intelligence capability identify
 and is looking for the detectable thing
  'wheeled vehicle' and
 provides the capability 'visible sensing' and
 is rated as 'visible NIIRS rating 4'.
```

NIIRS ratings are associated with platform and sensor types, by means of the `provides` relationship (as with the `mounts` example above, instances of the `provides` relationship are defined between "prototypical" instances of the relevant sensor and platform types):

```
there is an EO camera named
  'EO camera sensor type' that
 provides the capability 'visible sensing'.

there is a Predator A named
 'Predator A platform type' that
 is rated as the NIIRS rating
  'visible NIIRS rating 6' and
 is rated as the NIIRS rating
  'RADAR NIIRS rating 4'.
```

4.4 Assignment Representation in CE

Entities associated with the KB lookup table described in Sect. 2.3 are modelled as follows:

```
conceptualise the assignment template AT
~ fulfills ~ the intelligence capability IC and
~ is looking for ~ the detectable thing DT and
~ can be satisfied by ~ the bundle type BT and
~ is ranked by ~ the utility function UF.
```

```
conceptualise the bundle type BT
~ is deployed on ~ the platform type P and
~ uses ~ the sensor type S.
```

Here is an example assignment template (and associated instances) in CE, based on the earlier examples:

```
there is an assignment template named at34 that
 fulfills the intelligence capability identify
 and is looking for the detectable thing
  'wheeled vehicle' and
 can be satisfied by the bundle type bt312 and
 is ranked by the utility function CDP.

there is a bundle type named bt312 that
 is deployed on the platform type
  'Predator A platform type' and
 uses the sensor type
  'EO camera sensor type'.

there is a utility function named CDP.
Note: CDP = Cumulative Detection Probabilty.
```

In our earlier ontology, a property *allocatedTo* was used to link tasks to assets. This proved inadequate to handle the richer relationship between asset bundles and tasks. For this purpose, we define the concept of an `assignment` as follows:

```
conceptualise an ~ assignment ~ A that
 has the task T as ~ task ~ and
 has the bundle B as ~ bundle ~ and
 has the value US as ~ utility score ~.

conceptualise the assignment A
 ~ uses ~ the assignment template AT and
 ~ is provided by ~ the coalition partner CP
  and
 ~ is owned by ~ the user UO and
 ~ is joined by ~ the user UJ.
```

The main relationships between `assignment` and other concepts are shown graphically in Fig. 7. Thus, an assignment is a concept that relates a task to a bundle and has an associated utility score. An assignment uses an assignment template (which defines the utility function used to generate the utility score). These definitions use three additional concepts: a sensor bundle (that conforms to a particular bundle

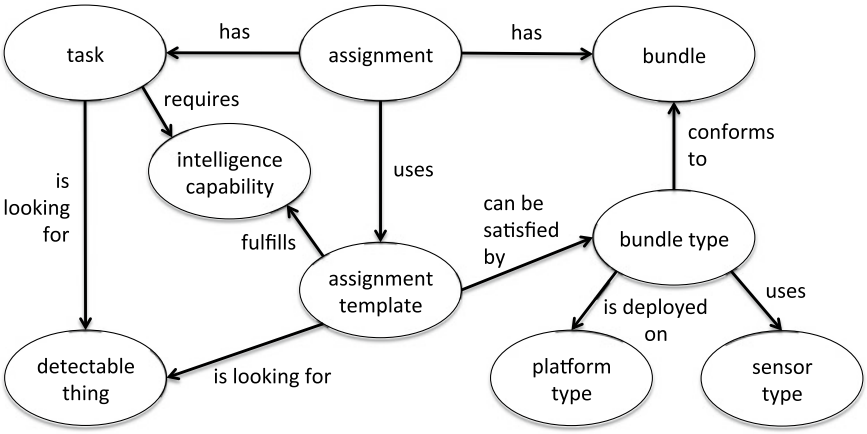


Fig. 7 Extract of model showing main relationships between `assignment` and linked concepts

type[13]), a user (individual person, for example an ISR analyst) and a coalition partner (which has ownership of particular assets). Incomplete definitions for these are shown below:

```
conceptualise the bundle B
 ~ conforms to ~ the bundle type BT.

conceptualise a ~ user ~ U.

conceptualise a ~ coalition partner ~ CP.
```

Assignments are generated by the procedure outlined in Sect. 2.3. An example assignment is shown below:

```
there is an assignment named a43288 that
 has the task t1265 as task and
 has the bundle b17352 as bundle and
 has '0.7' as utility score and
 uses the assignment template at34.

there is a bundle named b17352 that
 conforms to the bundle type bt312.
```

We also associate "provenance" information with an assignment, in terms of the coalition partner (typically a country) that provides the assets in the bundle, and the

[13] A bundle also contains specific asset instances, not shown here.

user who originated the task (the task "owner"). As described in the next section, we allow assignments to be shared among users—for this purpose, we define the relationship "is joined by". Some example provenance and sharing information on an assignment is shown below:

```
the assignment a43288
 is provided by the country UK and
 is owned by the user Sue41 and
 is joined by the user Bill356 and
 is joined by the user Tommy9 and
 is joined by the user Zack99.
```

Figure 7 can be seen as an extension of our original ontology depicted in Fig. 5, emphasising concepts of importance in the task-asset assignment process. Originally, there was no need to explicitly model the new concepts in Fig. 7 because most of them were internal to the assignment procedure. Having them made explicit allows us to make the assignment process more transparent, as per our second goal stated at the end of Sect. 1. In view of this, the next section shows how the CE knowledge base and instance representations can be used in a tool to facilitate task-asset assignment and sharing.

5 Task-Asset Assignment and Sharing Using a Tablet-Based App

Having established the feasibility of using CE to express our task-asset assignment knowledge base and instances, we now consider how the user in the field may exploit the capabilities afforded by our knowledge-based system. We created a client interface, implemented as an app on a tablet device, with the following main features:

1. Allow a user to create an ISR task in an area of interest, by means of a convenient user interface, and submit the task for asset assignment.
2. Achieve a separation between *what* information the user requires and *how* the information is obtained (by which kind of sensing).
3. Allow a user to view all tasks with assigned assets in an area of interest (subject to access policies).
4. Allow the sharing of tasks among users (again, subject to access policies).

The primary motivation for feature (3) was to give a decision-maker an overview of how well covered an area is in ISR terms, not just in terms of what tasks are currently being resourced, but also the likely permanence of these tasks (by allowing the user to view details of the takes such as their ownership and priority, which affect the likelihood of a task being pre-empted). The motivation for feature (4) was to reduce task pre-emption, by making it easier for a user to share an existing task than

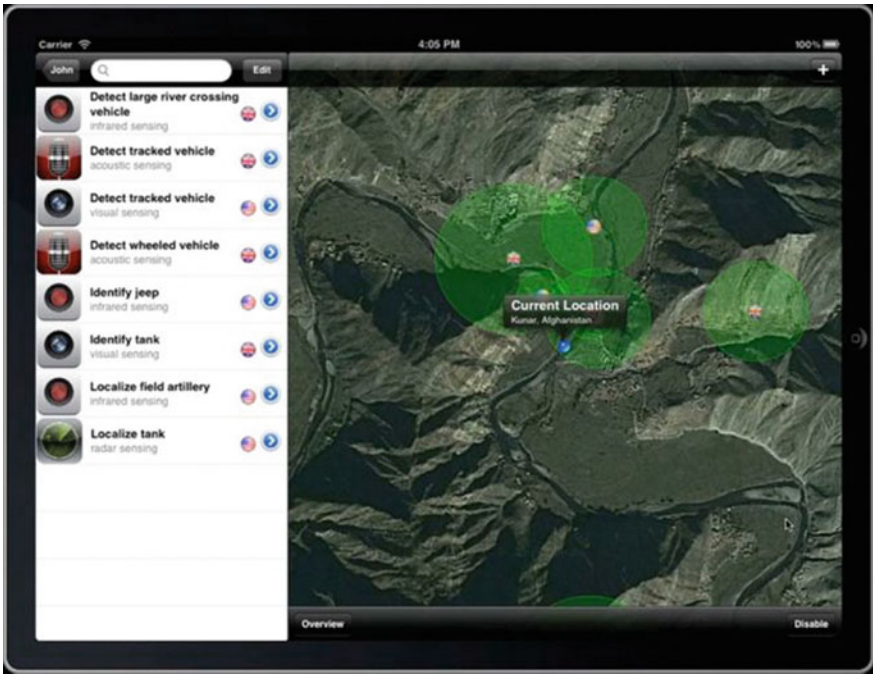


Fig. 8 Sensor assignment to missions (SAM) iPad app

to create a new request for resources (because tasks are only pre-empted when there is a competing task in the same area of interest).

Figure 8 shows the main panel for the iPad app: the *task panel*, illustrating the implementation of feature (3). The task list is on the left, and the locations of tasks are shown as circular regions on the map to the right. The list of tasks is contextualised by the area of interest, which is by default determined by the iPad's location (via GPS) but can be set manually also. This display is from the viewpoint of a logged-in user: each user belongs to a single coalition partner and sees a list of tasks visible according to their access policies. Access policies are rule-based and can take account of factors including the user's coalition partner membership, their rank, membership of a particular group within the coalition, and also the partner ownership, rank and group associated with the task.

The task list is searchable using the box on the top-left allowing users to filter the displayed list by intelligence capability or type of detectable thing. Tasks are summarised view with an icon that provides a quick indication of the kind of sensing capability assigned (e.g. by means of acoustic, visual, infrared, seismic or radar sensing). Information for this list can be obtained by querying the CE knowledge base for `assignment` instances. As noted above, the `capability` is displayed both in text and as an icon. The `coalition partner` is displayed as a "flag" icon to the right of each item and also at the centre of the task on the map. (These

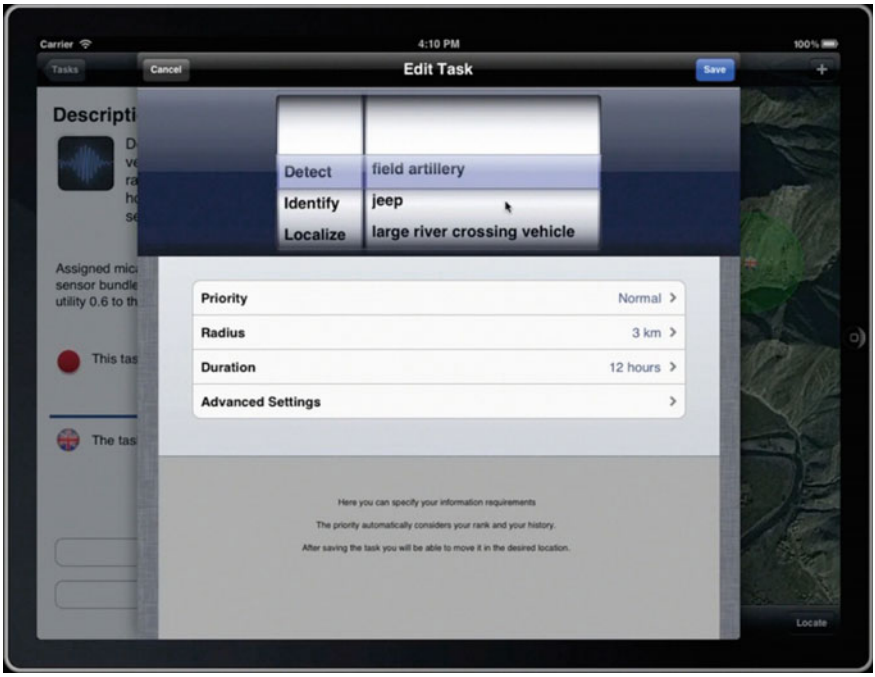


Fig. 9 iPad app task input form

could be more fine-grained than country-level—the flags are merely for illustrative purposes.)

A user may create an ISR task using the form shown in Fig. 9, corresponding to the task model described in Sect. 4.3. The app allows a user to specify the area of interest of the task in terms of a point on the map and radius.

The task panel on the left of Fig. 8 shows only a summary of tasks. More detail on each can be obtained by selecting an individual task, which results in an *assignment view* like the one shown in Fig. 10. Assuming the task has assigned sensors, the display presents a text summary derived from the CE model. With this amount of detail, the user can understand how the task is currently being resourced. Inclusion of the NIIRS rating is intended to give an indication of the quality of the data the resourcing bundle can collect. We are considering other ways to convey quality information, as it may not be reasonable to assume users are familiar with the NIIRS scale. As noted above, the task priority is an indication of how likely the task is to be pre-empted (lower-priority tasks are more prone to this). Information on the task owner and other users who have joined the task is intended to have a "social" effect, as the logged-in user is likely to know other users in the region.

Based on this detailed task assignment information, the user may use the buttons on the bottom left to choose to join or edit the existing task. The intent here is that it is a more efficient use of network resources for a user to join an existing task than

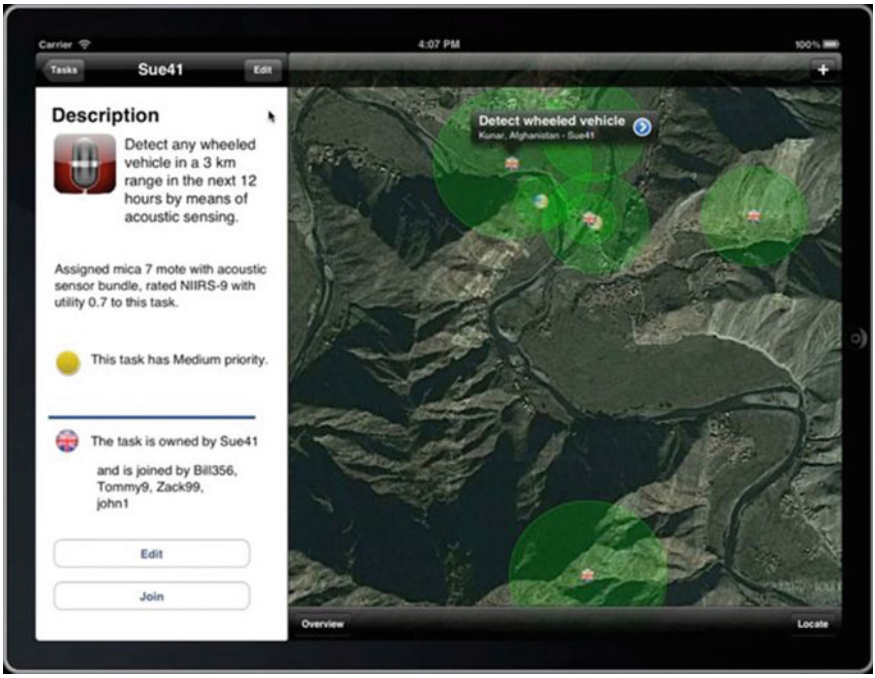


Fig. 10 Details of an assigned task

to create a new task which will compete for resources with existing tasks. Currently, editing an existing task effectively creates a new task (though is a quicker procedure for the user) though we are considering ways to avoid this in particular cases where the edited task can still be resourced by the currently assigned bundle.

In summary, as a result of the latest implementation work we have demonstrated that all information provided by the current user interface can be generated through processing of the underlying CE sentences as the direct underlying information representation format, without the need for conversion to a secondary form such as OWL. This is achieved through the use of the CE store processing environment which provides a set of APIs to define and query information in the form of CE sentences.

The iPad app was demonstrated to subject matter experts from the USA, UK and NATO communities. Highly positive feedback was obtained, especially concerning that features (3) and (4) for asset sharing are considered important for a realistic deployment of our approach.

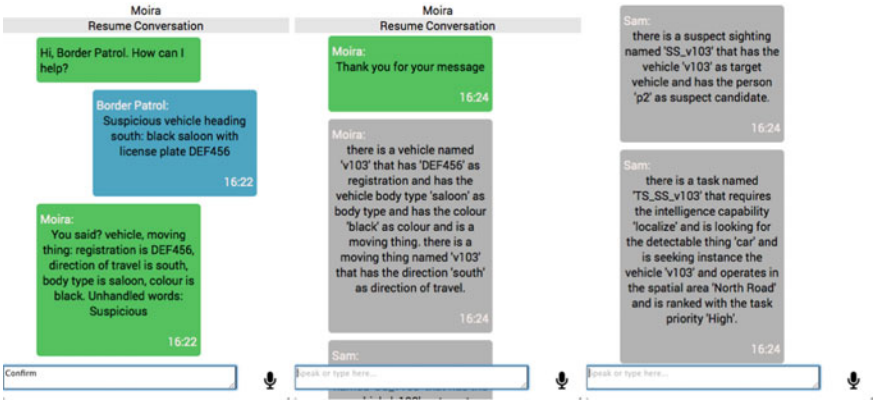


Fig. 11 A conversation with Moira and Sam agents using a prototype smartphone interface

5.1 Prototype Conversational Agents

To illustrate the conversational D2D concept, prototype conversational agents have been implemented and tested in limited experiments [4, 22]. Two distinct agent functionalities have been identified as useful and reusable:

- A conversational agent whose main purpose is to mediate interactions with human users (mainly confirm and gist/expand). This agent is called **Moira** (Mobile Intelligence Reporting App).
- A conversational agent whose main purpose is to apply knowledge of tasks and ISR assets to match tasks to available sensing assets. This agent uses the SAM algorithm presented earlier in the chapter and is therefore named **Sam**.

One interface to the Moira agent, deployed via a smartphone, is shown in Fig. 11. The sequence of screenshots depicted here reflects the three use cases described above. The smartphone user (whose name is "Border Patrol") interacts with Moira by speech or typing. Their messages are shown in blue. Moira's messages are in green. In this case, the user is also permitted to see other conversations in which Moira is involved (shown in grey), so they see the exchange between Moira and Sam that initiates the new task request to track the vehicle. Note that the form of the confirmatory message shown in the second green bubble in the leftmost screenshot uses a gist form rather than full CE, for the reasons given above (brevity and low complexity).

A pilot implementation of Moira has also been created for an eyeline-mounted display such as Google Glass.[14] Early experiments suggest a gist form of confirmatory message is even more appropriate here. An example of this is shown in Fig. 12 where the user sees a combination of machine-generated images and compact text. In general, the style and format (e.g. text and/or graphics) chosen by Moira for

[14] http://glass.google.com.

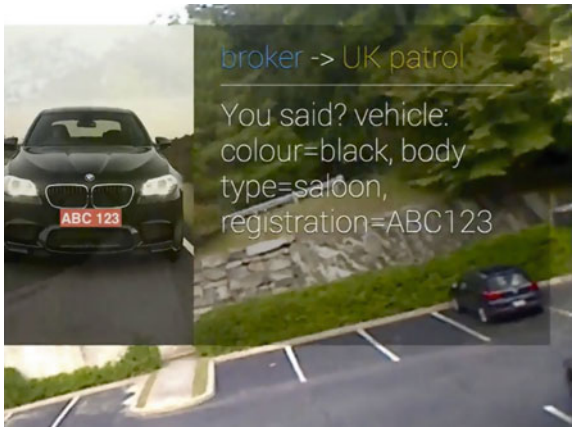


Fig. 12 Experimentation with a graphical form for confirmatory messages

confirm and gist/expand interactions can be based on additional contextual factors such as the user, their role, the current operational tempo and the form factor of the device they are using.

6 Conclusion

In this chapter, we set out to describe the sensor assignment to missions (SAM) approach automated asset-task matching in an ISR context. We reviewed the specification of the SAM algorithm as a logic program and reformulated our original model (based on the Military Missions and Means Framework) into Controlled English (CE) with no loss of power to support our automated asset-task matching procedure. We showed how the new model can be used in the context of a user-facing mobile app to assist understanding of the currently resourced set of ISR tasks and informed choices on whether to create a new task or share an existing task.

CE provides a good basis for translation between human languages and may offer advantages in a coalition context. Our tablet app demonstrates how a multi-modal interface can be driven by underlying CE-based information, with the information being presented to the user in a manner that is conducive to their task, for example labelled points on a map rather than CE sentences with lat/long values. Because the system is based on CE, if a user wishes to go outside the designed scope of the application, they can contribute CE sentences that conform to the underlying model in order to interact with the system if required.

We believe that the current form of CE provides a good start and does appeal to non-technical users, but we wish to invest further into the refinement of the syntax and an increased expressivity to allow information to be expressed more eloquently without sacrificing the underlying machine processability. Since the underlying CE

model is extensible the opportunity for "local knowledge" to be added via the app is very real (both in terms of new facts and in terms of model refinements/extensions).

Acknowledgements This research was sponsored by the US Army Research Laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defence or the UK Government. The US and UK Governments are authorised to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

I am indebted to the many colleagues and students with whom I have collaborated on the SAM approach, including Amotz Bar-Noy, Konrad Borowiecki, Dave Braines, Mario Gomez, Chris Gwilliams, Tom La Porta, Matt Johnson, Geeth de Mel, Gavin Pearson, Tien Pham, Diego Pizzocaro and Hosam Rowaihy.

Appendix

This code has been tested in SWI Prolog[15] 5.10.4 for MacOSX 10.7.

```
% NIIRS table generation procedure

:- dynamic
  intTask/1,
  hasIntCapability/2,
  hasDetectables/2,
  hasAreaOfInterest/2,
  hasTimePeriod/2.

generateKBTable :-
  retractall( intTask( _ ) ),
  retractall( hasIntCapability( _, _ ) ),
  retractall( hasDetectables( _, _ ) ),
  retractall( kbTable( _, _, _ ) ),
  generateIstarTask( Task, IntCap, Detectables ),
  matches( Task, [ Platform | Sensors ], _ ),
  matchJointUtilityModel( IntCap, [ Platform | Sensors ],
    JointUtilityModel ),
  sort( Sensors, SortedSensors ),
  assertIfNew( [ IntCap, Detectables ],
    [ Platform | SortedSensors ],
    JointUtilityModel ),
  fail.
generateKBTable :-
```

[15] http://www.swi-prolog.org/.

```
    findall( [ Task, BundleType, JointUtilityModel ],
      kbTable( Task, BundleType, JointUtilityModel ), KBTable ),
    sort( KBTable, SortedKBTable ),
    writeKBTable( SortedKBTable ).

generateIstarTask( T, IntCap, Detectables ) :-
    intTaskType( IntCap, Detectables ),
    gensym( t, T ),
    assertz( intTask( T ) ),
    assertz( hasIntCapability( T, IntCap ) ),
    assertz( hasDetectables( T, Detectables ) ).

intTaskType( IntCap, [ Detectable ] ) :-
    intCapability( IntCap ),
    IntCap \= distinguish,
    transitiveSubClassOf( Detectable, detectable ).
intTaskType( distinguish, SortedDetectables ) :-
    transitiveSubClassOf( Detectable1, detectable ),
    transitiveSubClassOf( Detectable2, detectable ),
    Detectable1 \= Detectable2,
    Detectables = [ Detectable1, Detectable2 ],
    sort( Detectables, SortedDetectables ).

% Matchmaker predicates

matches( Task, PlatformConfig, Capabilities ) :-
    requiresCapabilities( Task, Capabilities ),
    platformConfiguration( PlatformConfig ),
    providesCapabilities( PlatformConfig, Capabilities ),
    minimalConfiguration( PlatformConfig, Capabilities ).

requiresCapabilities( Task, [ IntType, Rating ] ) :-
    intTask( Task ),
    hasIntCapability( Task, IntCap ),
    hasDetectables( Task, Detectables ),
    intClause( IntCap, DetectableSet, _, _, IntType, Rating ),
    subset( Detectables, DetectableSet ).

platformConfiguration( PlatformConfig ) :-
    transitiveSubClassOf( Platform, platform ),
    findall( Sensor, carries( Platform, Sensor ), Sensors ),
    subseq0( Sensors, SensorSubset ),
    SensorSubset \= [],
    isConfigurable( Platform, SensorSubset ),
    append( [ Platform ], SensorSubset, PlatformConfig ).
```

```
isConfigurable( Platform, Sensors ) :-
  subseq0( Sensors, [ Sensor1, Sensor2 ] ),
  unconfigurable( Platform, Sensor1, Sensor2) , !,
  fail.
isConfigurable( _, _ ).

unconfigurable( Platform, Sensor1, Sensor2 ) :-
  interferesWith( Platform, Sensor1, Sensor2 ).
unconfigurable( _, Sensor1, Sensor2 ) :-
  interferesWith( Sensor1, Sensor2 ).

providesCapabilities( PlatformConfig,
    [ Capability | Capabilities ] ) :-
  capabilityProvidedByPlatformConfig( Capability,
    PlatformConfig ),
  providesCapabilities( PlatformConfig, Capabilities ).
providesCapabilities( _, [] ).

capabilityProvidedByPlatformConfig( Capability,
    [ Asset | _ ] ) :-
  providesCapability( Asset, Capability ), !.
capabilityProvidedByPlatformConfig( Capability,
    [ _ | Assets ] ) :-
  capabilityProvidedByPlatformConfig( Capability, Assets ).

minimalConfiguration( [ Platform | Sensors ],
    Capabilities ) :-
  subseq0( Sensors, SensorSubset ),
  Sensors \= SensorSubset,
  providesCapabilities( [ Platform | SensorSubset ],
    Capabilities ), !,
  fail.
minimalConfiguration( _, _ ).


% Utility predicates

assertIfNew( Task, BundleType, JUM ) :-
  kbTable( Task, BundleType, JUM ), !.
assertIfNew( Task, BundleType, JUM ) :-
  assertz( kbTable( Task, BundleType, JUM ) ).

writeKBTable( [] ).
writeKBTable( [ [ Task, BundleType, JUM ] | KBTable ] ) :-
```

```
  write_ln( [ Task, BundleType, JUM ] ),
  writeKBTable( KBTable ).

subseq0( List, List ).
subseq0( List, Rest ) :- subseq1( List, Rest ).
subseq1( [ _ | Tail ], Rest ) :- subseq0( Tail, Rest ).
subseq1( [ Head | Tail ], [ Head | Rest ] ) :-
  subseq1( Tail, Rest ).


% sensor & platform ontology

transitiveSubClassOf( X, Y ) :-
  subClassOf( X, Y ).
transitiveSubClassOf( X, Y ) :-
  subClassOf( X, Z ),
  transitiveSubClassOf( Z, Y ).

carries( maleUAV, eoCamera ).
carries( maleUAV, tvCamera ).
carries( maleUAV, flirCamera ).
carries( maleUAV, ladar ).
carries( packbot510, tvCamera ).
carries( packbot510, flirCamera ).
carries( packbot510, acousticSensor ).
carries( micaZmote, acousticSensor ).
carries( micaZmote, magneticSensor ).
carries( Platform, Asset ) :-
  transitiveSubClassOf( Platform, ParentPlatform ),
  carries( ParentPlatform, Asset ).

interferesWith( eoCamera, flirCamera ).
interferesWith( packbot510, tvCamera, flirCamera ).

subClassOf( sensor, asset ).
subClassOf( platform, asset ).

subClassOf( eoCamera, sensor ).
subClassOf( tvCamera, sensor ).
subClassOf( flirCamera, sensor ).
subClassOf( acousticSensor, sensor ).
subClassOf( magneticSensor, sensor ).

subClassOf( predatorA, maleUAV ).
subClassOf( reaper, maleUAV ).
```

```
subClassOf( maleUAV, uav ).
subClassOf( packbot510, smallUGV ).
subClassOf( smallUGV, ugv ).
subClassOf( micaZmote, mote ).

subClassOf( aerialPlatform, platform ).
subClassOf( groundPlatform, platform ).
subClassOf( uav, aerialPlatform ).
subClassOf( trackedVehicle, groundPlatform ).
subClassOf( militaryTrackedVehicle, trackedVehicle ).
subClassOf( heavyMilitaryTrackedVehicle,
  militaryTrackedVehicle ).
subClassOf( tank, heavyMilitaryTrackedVehicle ).
subClassOf( t62MBT, tank ).
subClassOf( largeRiverCrossingVehicle,
  heavyMilitaryTrackedVehicle ).
subClassOf( mediumMilitaryTrackedVehicle,
  militaryTrackedVehicle ).
subClassOf( wheeledVehicle, groundPlatform ).
subClassOf( car, wheeledVehicle ).
subClassOf( van, wheeledVehicle ).
subClassOf( jeep, wheeledVehicle ).
subClassOf( lorry, wheeledVehicle ).
subClassOf( militaryWheeledVehicle,
  trackedVehicle ).
subClassOf( missileSupportVehicle,
  militaryWheeledVehicle ).
subClassOf( sS25MobileMissileTEL,
  militaryWheeledVehicle ).
subClassOf( thermallyActivesSS25MobileMissileTEL,
  militaryWheeledVehicle ).
subClassOf( militaryGroundPlatform, groundPlatform ).
subClassOf( fieldArtillery, militaryGroundPlatform ).
subClassOf( train, groundPlatform ).
subClassOf( stringOfCarriages, groundPlatform ).
subClassOf( thermallyActiveVehicle, groundPlatform ).
subClassOf( ugv, groundPlatform ).
subClassOf( mote, groundPlatform ).

subClassOf( platform, detectable ).


% NIIRS KB
```

```
intCapability( detect ).
intCapability( identify ).
intCapability( distinguish ).
intCapability( localize ).


% NOTE all acoustic clauses assume acoustic scale aligns
% with visible & radar
intClause( detect, [ wheeledVehicle ], [],
  [ motorpool ], radar, radar-4 ).
intClause( detect, [ wheeledVehicle ], [],
  [], acoustic, acoustic-4 ).
intClause( identify, [ wheeledVehicle ], [],
  [], visible, visible-4 ).
intClause( identify, [ trackedVehicle ], [],
  [], visible, visible-4 ).
intClause( identify, [ fieldArtillery ], [],
  [], visible, visible-4 ).
intClause( identify, [ largeRiverCrossingVehicle ], [],
  [], visible, visible-4 ).
intClause( detect, [ tank ], [],
  [ revetment ], infrared, infrared-5 ).
intClause( distinguish, [ wheeledVehicle, tank ], [],
  [], radar, radar-6 ).
intClause( distinguish, [ t62MBT, sS25MobileMissileTEL ],
  [], [], radar, radar-5 ).
intClause( detect, [ trackedVehicle ], [],
  [], visible, visible-4 ).
intClause( detect, [ trackedVehicle ], [],
  [], radar, radar-4 ).
intClause( detect, [ trackedVehicle ], [],
  [], infrared, infrared-5 ).
intClause( detect, [ trackedVehicle ], [],
  [], acoustic, acoustic-5 ).
intClause( identify, [ jeep ], [],
  [], visible, visible-6 ).
intClause( identify, [ jeep ], [],
  [], radar, radar-6 ).
intClause( identify, [ jeep ], [],
  [], infrared, infrared-7 ).
intClause( identify, [ jeep ], [],
  [], acoustic, acoustic-7 ).
intClause( identify, [ tank ], [],
  [], visible, visible-4 ).
intClause( identify, [ tank ], [],
```

```
    [], radar, radar-5 ).
intClause( identify, [ tank ], [],
    [], acoustic, acoustic-5 ).

intClause( detect, [ X ], _, _, IntType, Rating ) :-
    transitiveSubClassOf( X, Y ),
    intClause( detect, [ Y ], _, _, IntType, Rating ).

intClause( localize, [ X ], _, _, IntType, Rating ) :-
    intClause( detect, [ X ], _, _, IntType, Rating ).

providesCapability( acousticSensor, acoustic ).
providesCapability( tvCamera, visible ).
providesCapability( eoCamera, visible ).
providesCapability( flirCamera, infrared ).
providesCapability( ladarSensor, radar ).
providesCapability( predatorA, visible-6 ).
providesCapability( predatorA, radar-4  ).
providesCapability( reaper, visible-9 ).
providesCapability( reaper, radar-6  ).
providesCapability( reaper, infrared-6  ).
providesCapability( packbot510, acoustic-9 ).
providesCapability( packbot510, radar-6 ).
providesCapability( packbot510, visible-5 ).
providesCapability( micaZmote, acoustic-9 ).

providesCapability( Asset, Capability ) :-
    entailsCapability( Capability1, Capability ),
    providesCapability( Asset, Capability1 ).
providesCapability( Asset, Capability ) :-
    transitiveSubClassOf( Asset, ParentAsset ),
    providesCapability( ParentAsset, Capability ).

entailsCapability( IntType-Rating1,
      IntType-Rating2 ) :-
    ground( Rating1 ), !, Rating1 > 0,
    Rating2 is Rating1 - 1.
entailsCapability( IntType-Rating1,
      IntType-Rating2 ) :-
    ground( Rating2 ), !, Rating2 < 9,
    Rating1 is Rating2 + 1.
entailsCapability( IntType-Rating1,
      IntType-Rating2 ) :-
    between( 1, 9, Rating1 ), Rating2 is Rating1 - 1.
```

```
matchJointUtilityModel( localize, _,
  twoDimensionalLocalization ) :- !.
matchJointUtilityModel( _, [ _ | Sensors ],
    twoDimensionalLocalization ) :-
  member( acousticSensor, Sensors ).
matchJointUtilityModel( _ , _,
  cumulativeDetectionProbability ).
```

References

1. Bakdash, J.Z., Pizzocaro, D., Preece, A.: Human factors in intelligence, surveillance, and reconnaissance: gaps for soldiers and technology recommendations. In: Proceedings of the MILCOM (2013)
2. Bermudez, L., Graybeal, J., Arko, R.: A marine platforms ontology: experiences and lessons. In: Proceedings of the ISWC 2006 Workshop on Semantic Sensor Networks. Athens GA, USA (2006)
3. Botts, M., Robin, A.: OpenGIS sensor model language (SensorML) implementation specification. Technical report, Open Geospatial Consortium Inc. (2007)
4. Braines, D., de Mel, G., Gwilliams, C., Parizas, C., Pizzocaro, D., Preece, A.: Agile sensor tasking for CoIST using natural language knowledge representation and reasoning. In: Proceedings of the Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR V (SPIE), vol. 9079. SPIE (2014)
5. Broome, B.: Data-to-decisions: a transdisciplinary approach to decision support efforts at ARL. In: Proceedings of the Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR III (SPIE), vol. 8389. SPIE (2012)
6. Dumbill, E. (ed.): Planning for Big Data. O'Reilly (2012)
7. Fu, W.T., Gray, W.D.: Suboptimal tradeoffs in information seeking. Cogn. Psychol. **52**, 195–242 (2006)
8. Geyik, S., Szymanski, B., Zerfos, P.: Robust dynamic service composition in sensor networks. IEEE Trans. Serv. Comput. **6**(4), 560–572 (2013)
9. Goldstein, D.G., Gigerenzer, G.: Models of ecological rationality: the recognition heuristic. Psychol. Rev. **109**, 75–90 (2002)
10. Gomez, M., Preece, A., Johnson, M., de Mel, G., Vasconcelos, W., Gibson, C., Bar-Noy, A., Borowiecki, K., Porta, T.L., Pizzocaro, D., Rowaihy, H., Pearson, G., Pham, T.: An ontology-centric approach to sensor-mission assignment. In: Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2008), pp. 347–363. Springer (2008)
11. Guo, B., Wang, Y., Smart, P., Shadbolt, N., Nixon, M., Damarla, T.: Approaching semantically-mediated acoustic data fusion. In: Proceedings of the IEEE MILCOM) (2007)
12. Hall, C.C., Ariss, L., Todorov, A.: The illusion of knowledge: when more information reduces accuracy and increases confidence. Organ. Behav. Hum. Decis. Process. **103**, 277–290 (2007)
13. Irvine, J.M.: National imagery interpretability rating scales (NIIRS). In: Encyclopedia of Optical Engineering, pp. 1442–1456. Marcel Dekker (2003)
14. Laney, D.: 3D data management: controlling data volume, velocity, and variety. Technical report, META Group (2001)
15. Lefort, L., Henson, C., Taylor, K.: Semantic sensor network XG final report (2011). http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/

16. Llinas, J., Bowman, C., Rogova, G., Steinberg, A., Waltz, E., White, F.: Revisiting the JDL data fusion model II. In: Proceedings of the Seventh International Conference on Information Fusion (FUSION 2004), pp. 1218–1230 (2004)
17. McMullen, D., Bramley, R., Chiu, K., Davis, H., Devadithya, T., Huffman, J., Huffman, K., Reichherzer, T.: The common instrument middleware architecture: experiences and future directions. In: Grid Enabled Remote Instrumentation, pp. 393–407. Springer (2009)
18. de Mel, G., Sensoy, M., Vasconcelos, W., Preece, A.: Flexible resource assignment in sensor networks: a hybrid reasoning approach. In: 1st International Workshop on the Semantic Sensor Web (SemSensWeb 2009) (2009)
19. Mott, D.: Summary of ITA Controlled English (2010). https://www.usukita.org/papers/5658/details.html
20. Mullen, T., Avasarala, V., Hall, D.L.: Customer-driven sensor management. IEEE Intell. Syst. **21**(2), 41–49 (2006)
21. Pizzocaro, D., Preece, A., Chen, F., Porta, T.L., Bar-Noy, A.: A distributed architecture for heterogeneous multi sensor-task allocation. In: Proceedings of the 7th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'11) (2011)
22. Preece, A., Braines, D., Pizzocaro, D., Parizas, C.: Human-machine conversations to support multi-agency missions. ACM SIGMOBILE Mobile Comput. Commun. Rev. **18**(1), 75–84 (2014)
23. Preece, A., Gomez, M., de Mel, G., Vasconcelos, W., Sleeman, D., Colley, S., Pearson, G., Pham, T., Porta, T.L.: Matching sensors to missions using a knowledge-based approach. In: SPIE Defense Transformation and Net-Centric Systems 2008 (SPIE Defense and Security Symposium, SPIE Proceedings, vol. 6981). SPIE (2008)
24. Preece, A., Norman, T., de Mel, G., Pizzocaro, D., Sensoy, M., Pham, T.: Agilely assigning sensing assets to mission tasks in a coalition context. IEEE Intell. Syst. **Jan/Feb**, 57–63 (2013)
25. Preece, A., Pizzocaro, D., Borowiecki, K., de Mel, G., Vasconcelos, W., Bar-Noy, A., Johnson, M., Chen, F., Porta, T.L., Rowaihy, H.: Knowledge-driven agile sensor-mission assignment. In: Proceedings of the 3rd Annual Conference of the International Technology Alliance (ACITA 2009) (2011)
26. Preece, A., Webberley, W., Braines, D.: Tasking the tweeters: obtaining actionable information from human sensors. In: Proceedings of the Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR VI (SPIE), vol. 9464. SPIE (2015)
27. Rowaihy, H., Johnson, M., Pizzocaro, D., Bar-Noy, A., Kaplan, L., Porta, T.L., Preece, A.: Detection and localization sensor assignment with exact and fuzzy locations. In: Proceedings of the 5th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'09) (2009)
28. Russomanno, D., Kothari, C., Thomas, O.: Building a sensor ontology: a practical approach leveraging ISO and OGC models. In: Proceedings of the International Conference on Artificial Intelligence, pp. 637–643 (2005)
29. Sheehan, J.H., Deitz, P.H., Bray, B.E., Harris, B.A., Wong, A.B.H.: The military missions and means framework. In: Proceedings of the Interservice/Industry Training and Simulation and Education Conference, pp. 655–663 (2003)
30. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artif. Intell. **101**, 165–200 (1998)
31. Simonis, I., Echterhoff, J.: OGC Sensor Planning Service Implementation Standard. Technical report, Open Geospatial Consortium (2011). http://www.opengis.net/doc/IS/SPS/2.0
32. Sowa, J.: Common Logic Controlled English (2004). http://www.jfsowa.com/clce/clce07.htm
33. W3C: SWRL: A semantic web rule language combining OWL and RuleML. World Wide Web Consortium (2004). http://www.w3.org/Submission/SWRL/
34. W3C: OWL 2 Web ontology language document overview, 2nd edn. World Wide Web Consortium (2012). http://www.w3.org/TR/owl2-overview/

Resource Allocation and Task Scheduling in the Cloud of Sensors

Igor L. dos Santos, Flávia C. Delicato, Luci Pirmez, Paulo F. Pires and Albert Y. Zomaya

Abstract The cloud of sensors (CoS) paradigm has emerged from the broader concept of Cloud of Things, and it denotes the integration of clouds and wireless sensor and actuator networks (WSANs). By integrating clouds with WSAN, some tasks initially assigned to smart sensors can be off-loaded to the cloud, thus benefiting from the huge computational capacity of these platforms. However, for time-critical applications, the high and unstable latency between the sensors and the cloud is not desirable. Besides low latency, WSAN applications usually require mobility and location-awareness properties, not often supported by current cloud platforms. Moreover, the indiscriminate off-loading of data/tasks from sensors to the cloud may lead to an overutilization of the network bandwidth while, in some cases, sensor-generated data could be locally processed and immediately discarded. To overcome these drawbacks of the integration between WSANs and the cloud, the edge paradigm emerges as a promising solution. Edge computing refers to enabling the computing directly at the edge of the network (for instance, through smart gateways and micro-data centers). Combining the paradigms of cloud/edge computing and WSANs in a three-tier architecture potentially leverages mutual advantages while posing novel research challenges. One of such challenges regards the development of solutions for performing resource allocation and task scheduling for CoS. Both edge and cloud paradigms strongly rely on the virtualization of physical resources. Therefore, resource allocation in CoS refers to the process of allocating instances of virtual nodes to perform the application requests (workload) submitted to the CoS, trying to meet as best as possible the requirements of applications, while respecting the constraints of the underlying physical infrastructure. Task scheduling denotes the process of selecting a group of physical nodes that are suitable for the execution, in a given order, of the various tasks necessary to meet an application request. The goal of this chapter is to

I. L. dos Santos · F. C. Delicato (✉) · L. Pirmez · P. F. Pires
Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
e-mail: fdelicato@dcc.ufrj.br

A. Y. Zomaya
School of Information Technologies, Centre for Distributed and High Performance Computing, University of Sydney, Sydney, Australia

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_8

overview the state of the art in the development of solutions for these two essential activities for the construction and efficient execution of CoS infrastructures.

1 Introduction

During the first decades of the twenty-first century, we have been witnessing the wide spread of a multitude of new paradigms, which are revolutionizing the field of Information and Communication Technologies [1]. One of such paradigms is the Internet of Things (IoT). The currently accepted definition for IoT envisions a global network infrastructure, linking a wide variety of physical and virtual devices, the so-called smart things, which are endowed with identification, data processing, sensing, and connection capabilities for supporting the development of cooperative services and applications [2].

IoT devices (the smart things) are highly heterogeneous, varying from wearables to sensor-instrumented vehicles, and they can be connected to the Internet via wired or wireless links and using different communication protocols. Among the highly heterogeneous set of smart things, the smart sensors have been playing an important role in the IoT paradigm [3]. Smart sensors are tiny battery-powered devices, endowed with processing, storage, sensing, actuation, and wireless communication capabilities. The communication capability enables grouping smart sensors to compose a Wireless Sensor and Actuator Network (WSAN) [4]. Besides sensor nodes, WSANs in general include sinks, which are nodes without the resource constraints (in terms of computation, energy, and communication) typical of smart sensors. Sink nodes are linked to gateways, thus enabling the connection between the WSAN and external systems/networks, such as the Internet. Multiple WSANs integrated through and with the Internet serve as the underpinning for building the IoT ecosystem, as shown in Fig. 1.

When dealing with the global scale, the sharing of physical WSANs among multiple applications will be the typical case, and the increasing amount of data generated by the sensors will require high computational power to provide useful and timely services for the end user [3]. Thus, a growing interest has emerged in the literature regarding the combination of WSANs with cloud computing [5]. A cloud may host several virtual instances called cloud nodes (CN), which are instantiated from and run on typical physical cloud data centers, and provide to users a large pool of resources. Cloud platforms, with their abundant resources, come naturally hand-in-hand with WSANs to create complex, large-scale, distributed, and data-oriented ecosystems.

By integrating clouds with WSAN, some tasks initially assigned to smart sensors can be offloaded to the cloud. However, some time-sensitive tasks may require fast response. In such cases, a long and unstable latency between the smart sensors and the cloud is not desirable [6]. Besides low latency, WSAN applications usually require mobility and location-awareness support [7], which clouds often do not implement. Finally, the indiscriminate offloading of data/tasks from sensors to the cloud may lead to an overutilization of the network bandwidth while, in some cases, most of

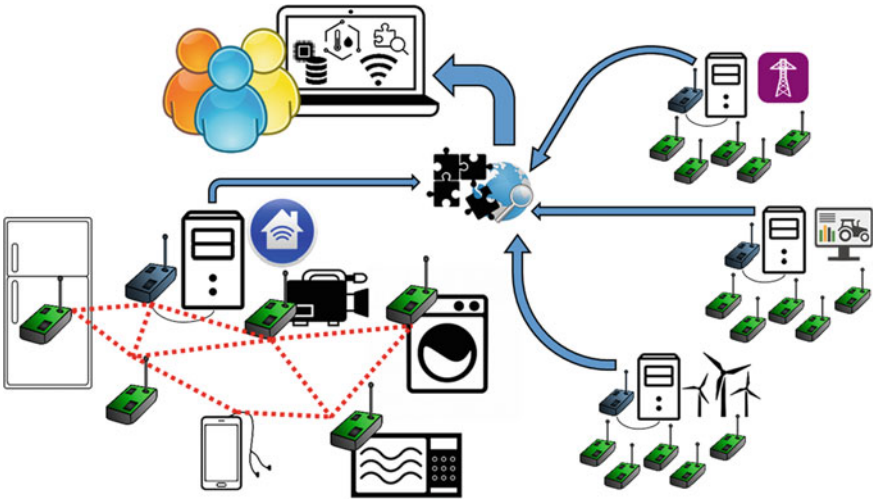


Fig. 1 WSANs enabling the IoT ecosystem

the produced data could be immediately and locally processed and discarded. To overcome these drawbacks of the integration between WSANs and the cloud, the edge computing [6–8] paradigm emerges as a promising solution.

Edge computing refers to enabling the computing directly at the edge of the network. As cloud computing, edge computing is a paradigm that proposes the virtualization of physical devices in the form of virtual instances, called edge nodes (EN) [9]. The edge nodes are instantiated from and run on physical edge devices to provide end users with computation, storage, and networking capabilities at the edge of the clouds [9]. Typical physical edge devices can be resource-poor devices such as access points, routers, switches, base stations, and smart sensors, or resource-rich micro-data centers and machines, such as cloudlets [7]. The edge nodes perform a number of tasks. For instance, they collect data from smart sensors and perform preprocessing, filtering, and reconstruction of the data into a more useful form, uploading only the necessary data to the cloud. In addition, the edge nodes can monitor smart objects and sensors activities, checking their energy consumption, besides assuring the security and privacy of their collected data. Thus, edge computing extends the traditional cloud computing paradigm to the edge of the network and enables time-sensitive and location-aware processing of the data collected by sensors.

In face of the exposed features of WSANs and cloud/edge computing, combining the paradigms of cloud/edge computing and WSANs in a three-tier architecture potentially leverages mutual advantages. On the one hand, WSANs could benefit from the powerful computing resources, besides the low latency, mobility, and location-awareness support of a cloud/edge environment to implement service management and composition for exploiting the smart sensors and their produced data. On the other hand, the cloud/edge can benefit from WSANs by extending its scope to

deal with real-world objects in a distributed and dynamic way, enabling the delivery of new services in a wider range of real-world scenarios. Moreover, such three-tier architecture is able to support both delay-sensitive and computation-intensive WSAN applications, by allocating such applications to run on, respectively, the edge tier and the cloud tier. When exploiting the aforementioned synergies, a new paradigm emerges, called the Cloud of Sensors (CoS) [10].

The CoS infrastructures are built upon the concept of WSAN virtualization [11], which is expected to provide a clean decoupling between the applications and the physical WSAN infrastructure. By virtualizing the physical infrastructure of the WSAN, it is possible to share it between multiple applications simultaneously. At the same time, the specific requirements of each application can be used to guide the creation of virtual nodes or networks, thus enabling a cost-effective sharing and avoiding negative interferences between the various applications. In this sense, virtualization allows a single physical network to serve multiple missions (one per application) in order to increase the return of investment (ROI) on infrastructure (from the owner or provider's point of view). Thus, the vision of mission-oriented networks [12, 13] can be efficiently implemented, reconciling the need to tailor the WSAN to each specific mission on the one hand, and the need for optimizing the usage of the resources on the other hand.

Essentially, in the CoS paradigm, the cloud/edge acts as intermediate layers between the physical smart sensors and the applications. It is important to mention that not all the works described hereafter in this chapter consider the full three-tier architecture. Some of them adopt a two-tier design, including the cloud but excluding the edge tier. In either case, these intermediate layers hide the complexity of the physical infrastructure necessary to implement applications, facilitating their delivery and allowing the sharing of the physical infrastructure among several applications.

To better illustrate the motivation for WSAN sharing and interface with the cloud and edge, consider the following example of application scenario. The goal of this scenario is to monitor the structural health of a smart building, using the method described in our previous work [14]. Our previous work considered the typical approach of implementing a WSAN specifically for one application. Within the four floors of the smart building, the sensor nodes are deployed in key locations where the shock response of the structure is the highest. The sensor nodes collect and provide data to applications regarding the damage coefficient of the respective floor, as described in [14]. In addition, we consider several applications, based in [14], that request such data, in order to monitor the structural health of the building. Although such analysis can be carried out for the building, such system may collapse fast when either the amount of applications or the size of the analysis increases. For instance, if the infrastructure of this smart building is replicated for a whole city or country, an application user, working for the government, may be interested on the high-level visualization of the structural health of all buildings in a city. With the output data of several instances of the application previously described, each one running in a smart building, it is possible to build a full map of structural health, requiring the cloud node for complex processing and analytics. Therefore, besides sensor nodes, gateway nodes can be deployed, each one attached to a local desktop,

forming the edge nodes (one edge node per floor). The edge nodes connect all the floors through the building LAN. The LAN connects to the Internet by using existing cable or wireless broadband internet connections (for instance, ADSL, VDSL, Satellite) in the building, thus being able to reach the cloud node when necessary. According to our previous work [8], the edge tier is chosen as the best location in our three-tier CoS architecture to run the virtual nodes. The sensor nodes can forward their data to the edge tier, where such data will be shared among applications through the virtual nodes. Moreover, several other virtual nodes can be instantiated and applications can be composed from them, to compute a myriad of structural damage indicators for each building. Since a smart city has several smart buildings, each one with several floors, the data regarding structural damage indicators should not be transmitted integrally to the cloud. In line with the edge computing paradigm, we suggest the intensive use of the processing of virtual nodes at the edge tier, to reduce this data and responding with actuations faster.

Despite the potential advantages offered by the CoS, there are still several challenges to deal with in the design of such infrastructures, to fully enable the CoS paradigm. One of such challenges regards the development of solutions for performing resource allocation [15, 16] and task scheduling [17, 18] in the CoS environment. Hereafter, the goal of this chapter is to overview the state of the art in the development of solutions for this challenge. To achieve this goal, we organize this chapter as follows. Section 2 depicts the background concepts of resource allocation and task scheduling in CoS. In Sect. 3, we briefly describe several solutions found in the literature that we consider being of utmost importance for understanding the context of task scheduling and resource allocation in CoS. In Sect. 4, we briefly overview our own approach for resource allocation and task scheduling in CoS. Finally, Sect. 5 portrays the concluding remarks and draws future research directions, guiding further studies in resource allocation and task scheduling in CoS. We believe this chapter is a useful reference for everyone who wishes to start his/her studies in this novel domain of CoS.

2 Background Concepts

In this section, we provide further information about the CoS architecture (Sect. 2.1) and virtualization model (Sect. 2.2). Moreover, we provide formal definitions for task scheduling (Sect. 2.3) and resource allocation (Sect. 2.4) in CoS.

2.1 The CoS Architecture

In this work, we consider the CoS architecture shown in Fig. 2. Three tiers compose this architecture: sensors tier (ST), edge tier (ET), and cloud tier (CT).

The sensors tier comprises the physical wireless sensor and actuator network infrastructures (WSANIs), each of which is owned and administered by an infras-

tructure provider (InP). Each WSANI comprises a set of physical sensor and actuator nodes (PSANs) deployed over a geographical area and connected by wireless links, so that every PSAN pertains to a single WSANI. Per Eq. (1), we describe each PSAN in terms of processing speed PS, total memory TM, list of sensing units LS, list of actuation units LA, remaining energy EN, identification of the WSANI to which it pertains WI and information about its location coordinates LC.

$$psan_i = \langle PS, TM, LS, LA, EN, WI, LC \rangle \tag{1}$$

In addition, we define that S_PSAN, in Eq. (2), is the set of α PSANs existing in the sensors tier.

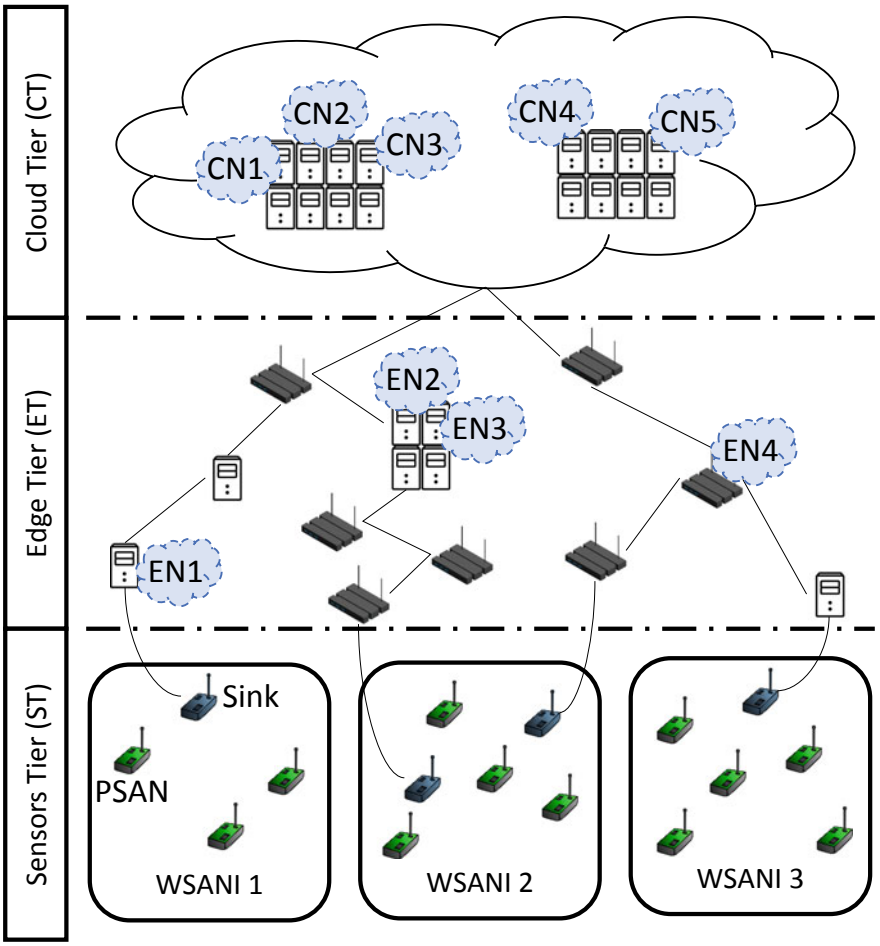


Fig. 2 CoS architecture

$$S_{PSAN} = \{psan_i | i \in \mathbb{N}_0 \, and \, i < \alpha\} \tag{2}$$

The InPs define the physical and administrative (logical) boundaries of their respective WSANIs. In our architecture, we assume that the physical boundaries of WSANIs (defined by the geographical area of deployment) may overlap, without influencing the administrative boundaries (defined by InP logic). Thus, the administrative boundaries may differ to the underlying physical boundaries, whose management is not in the scope of our work. Moreover, each PSAN has the knowledge of a valid communication path to reach the sink node within the WSANI. These communication paths are defined by underlying networking protocols chosen by the InP, such as [19, 20] and are not in the scope of our model.

The edge tier comprises the edge nodes, which are typical physical edge devices. Such devices can be resource-poor devices such as access points, routers, switches, base stations, and smart sensors, or resource-rich micro-data centers and machines, such as cloudlets [7]. Per Eq. (3), we describe each edge node in terms of processing speed PS, total memory TM, bandwidth BW, and its physical host identification HI.

$$en_i = \langle PS, TM, BW, HI \rangle \tag{3}$$

In addition, we define that S_EN, in Eq. (4), is the set of β edge nodes existing in the edge tier.

$$S\_EN = \{en_i | i \in \mathbb{N}_0 \, and \, i < \beta\} \tag{4}$$

The cloud tier comprises the cloud nodes, which are multiple physical data centers (more powerful than physical edge devices) responsible for a global view of the CoS system. Cloud physical data centers are able to perform data-intensive computation, time-based data analysis, and permanent storage of huge amounts of valuable data. The InPs can combine multiple physical data centers for providing services in the global scale, and each physical data center has a heterogeneous cost for providing its services. According to Eq. (5), we describe each cloud node in terms of processing speed PS, total memory TM, bandwidth BW, and its physical host identification HI.

$$cn_i = \langle PS, TM, BW, HI \rangle \tag{5}$$

Finally, we define that S_CN, in Eq. (6), is the set of γ cloud nodes existing in the cloud tier.

$$S\_CN = \{cn_i | i \in \mathbb{N}_0 \, and \, i < \gamma\} \tag{6}$$

In our CoS architecture, we consider the existence of three main entities, namely the PSANs at the sensors tier, the edge nodes at the edge tier, and the cloud nodes at the cloud tier. In our work, we choose to model PSANs as being the physical devices of the sensors tier themselves, while we consider the edge and cloud nodes as being virtual instances hosted by the physical devices of edge and cloud tiers, respectively.

This choice allows us to consider that the deployment of each edge and cloud node on their respective physical hosts is transparent to our CoS architecture and is handled by typical cloud and edge computing virtualization models. In addition, each edge and cloud node has information about the physical device that hosts it, so that we can associate the physical location of an edge or cloud node to the location of its physical host. In Sect. 2.2, we discuss issues on CoS virtualization, which is performed over the CoS architecture considered in this subsection.

2.2 CoS Virtualization

The capabilities of the CoS architecture, described in Sect. 2.1, are provided to users and their applications through a CoS virtualization model. Several works such as [11, 21–24] correlate the virtualization of physical devices in CoS to the resource allocation and task scheduling problems, suggesting that the design of solutions to these problems needs to be investigated together to enable the CoS virtualization itself. It is important to mention that by CoS virtualization, we mean the decoupling between the physical infrastructure of the WSAN and applications via an abstract representation of the former, with the main purposes of sharing the physical infrastructure of WSANs. In our point of view, a full CoS virtualization model comprises the instantiation of virtual nodes (VNs) and at least two sub-processes, one for performing resource allocation and the other for task scheduling. Figure 3 summarizes the relation between both resource allocation and task scheduling in CoS.

In the CoS paradigm, application users, with any given level of application domain expertise, define and implement applications. An application is similar to a workflow that describes interactions among the services provided by VNs. In addition, we call each activity of the application workflow as a request, because its main goal is to request (demand) the services of VNs. Thus, each application consists of a set of requests (activities of workflows) that demand the resources of the CoS infrastructure. A request is a set of abstract commands defined by users, which represent the application functional requirements, representing an abstract service. In addition, application requests have functional and non-functional requirements. Among the possible non-functional requirements, the data freshness is one of the most important in a distributed system based on sensing data, such as the CoS [25]. This importance increases particularly in the context of systems composed of a set of autonomous data sources, where the data integration with different freshness values can lead to semantic problems, hindering the execution of applications. There are several definitions of data freshness in literature [25]. Among these definitions stands out the one that quantifies the freshness of a given data based on the time elapsed since its acquisition.

Applications typically access the CoS through the edge and/or cloud tiers. During the operation of the CoS, several applications access the CoS, probably simultaneously, posing their requests. To meet such requests, the CoS infrastructure must provide the outputs (data, in case of sensing, or controls, in case of actuation) as

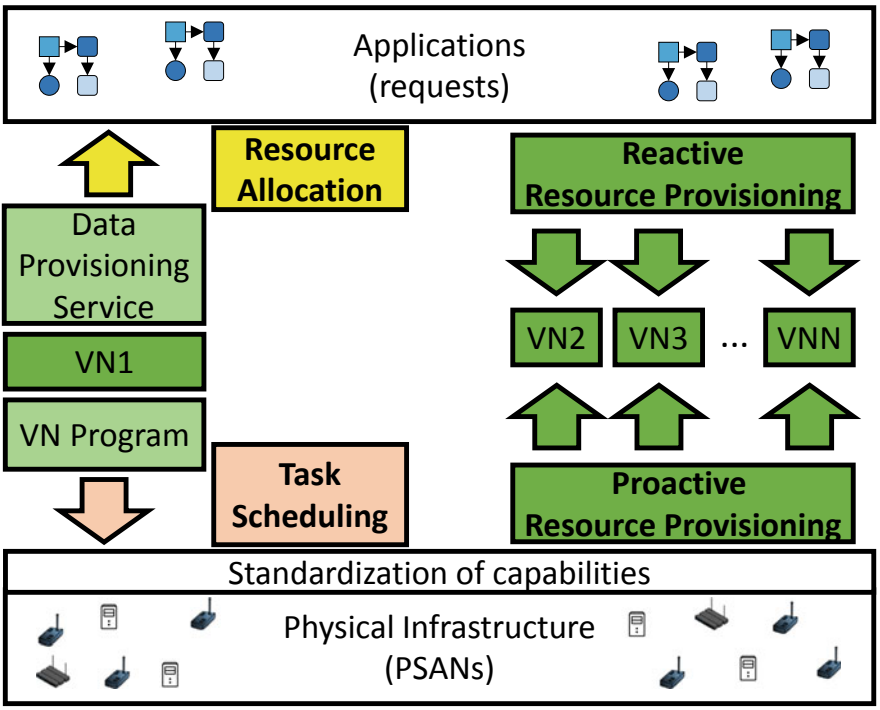


Fig. 3 Virtualization, resource allocation, and task scheduling in CoS

specified by the requests. Therefore, the CoS physical infrastructure has to perform tasks (generating a certain processing load on physical nodes) to provide such outputs with maximum data freshness. To avoid the re-execution of processing loads by the physical infrastructure, it is possible to meet other future requests by simply re-using outputs from previous executions, if they meet the data freshness requirement of the request. Moreover, these outputs could be stored at different tiers of the CoS architecture (sensors, edge, and cloud tiers). Therefore, it is necessary to make a first decision to meet a given request in the CoS environment: is it possible to meet the request using the data stored in the CoS environment (without engaging physical nodes), or is it necessary to dispatch tasks to run directly on the physical infrastructure? In the latter case, a second decision follows, regarding how to distribute the processing load among physical nodes that make up the CoS physical infrastructure. The first decision directly relates to the resource allocation process. Since this is a data acquisition process (either from real or virtual nodes), in our work we investigate it under the prism of data provisioning [26, 27]. The philosophy behind data provisioning consists of abstracting, from applications, data acquisition/access, and allowing data sharing among applications, while meeting respective application requirements. The second one is a task-scheduling decision. Traditional task scheduling algorithms in typical devices of the CoS [28] are responsible for selecting a group of physical nodes

that are suitable for the execution, in a given order, of the various tasks necessary to meet an application request.

In line with the concept of WSAN virtualization, the CoS system comprises the several VNs to perform the abstraction foreseen by data provisioning. Based on our previous work [10], throughout this chapter we define the VN as a computational entity (a software instance). The VN has the main goal of abstracting to users not only the data, but also the computation and communication capabilities provided by a set of underlying nodes. Thus, as a single virtual entity, the VN simplifies the representation of its underlying infrastructure to users. The underlying nodes are abstracted as services that the VN provides to applications. In the CoS, such software services mediate the interaction between applications and physical entities. Services expose resources, defined as software components that provide data from or control the actuation on physical entities [29]. In addition, the VN has the secondary goal of coordinating the execution of tasks by the underlying nodes, required to perform its first main goal. Thus, the implementation, the scheduling, and the execution of tasks in the physical infrastructure is the responsibility of VNs.

Therefore, we consider that every VN has a VN program, whose objective is to perform a series of tasks on the PSANs to update the data further provided by the VN through its data provisioning service. We consider that a VN program does not change during the existence of the VN. Thus, in CoS, task scheduling refers to the process, performed by the VN, of scheduling tasks of each VN program to a given set of nodes from the underlying physical infrastructure of the CoS. We consider the existence of a service at each VN that is responsible for running the task scheduling process. Moreover, we consider another service at each VN that implements the process of execution and supervision of the execution of the tasks, scheduled by the former process, on the underlying physical infrastructure. Similarly, a VN has a process for performing resource allocation and a process to execute applications requests on VNs allocated by the former process.

Finally, a process to instantiate the virtual nodes is first necessary, in order to select and prepare the underlying subordinated physical infrastructure for use. In the context of CoS, resource provisioning refers to this process of instantiating virtual nodes. The term "provisioning" often refers to the action of equipping or preparation of an infrastructure for some purpose. As in traditional cloud computing systems, the process of resource provisioning [30] is responsible for managing the association of physical computational capacities to counterpart virtual entities. This association, in the CoS, must maximize the utilization of physical computational (and data) capacities to virtual nodes, while respecting the physical computational (and data) capacities constraints of the physical CoS infrastructure. Based on [30], proactive resource provisioning denotes the cases where the instantiation of a VN occurs before the allocation of the VN to an application request. VNs are instantiated by the infrastructure providers at a time prior to the execution of the CoS infrastructure and the arrival of applications. In case of reactive resource provisioning, the instantiation of a VN may also occur (in addition to the reuse of existing instances) in response to the need of its allocation to an application request (i.e., the requirements of a new VN are tailored to meet a specific allocation). Thus, VNs are instantiated on

demand, by a dynamic (real-time) and automated resource provisioning algorithm. Generally, resource provisioning and adjustment according to demand should occur dynamically, in an elastic and transparent way.

It is important to mention that in our previous work [8], we concluded that the best theoretical position for positioning VNs is the edge tier. Among PSANs, edge nodes, and cloud nodes, we chose the edge nodes to run VNs for the following reasons. The edge nodes have greater computation and communication capabilities than PSANs and thus are able to manage VNs instances more efficiently. Moreover, edge nodes are in the edge of the network, closer to WSANIs and to the end users than the cloud nodes. Thus, edge nodes are in a privileged position, in relation to the cloud nodes, for linking PSANs from different WSANIs under the same VN, and for reducing the latency for time-sensitive applications. In addition, the cloud nodes store the registries of existing VNs in the whole CoS. Thus, we consider that edge nodes are the possible hosts for the VNs instantiated by the resource provisioning process.

As the result of the resource provisioning procedure, we formally define the VNs and the set of VNs. Per Eq. (7), we describe each VN in terms of its list of underlying PSANs LU, services description SE and its host EN identification HI.

$$vn_i = \langle LU, SE, HI \rangle \tag{7}$$

Per Eq. (8), we define a set S_VN containing all the θ VN instantiated by Infrastructure Providers (InPs).

$$S\_VN = \{vn_i | i \in \mathbb{N}_0 \; and \; i < \theta\} \tag{8}$$

In Sect. 2.3, we will discuss the process of task scheduling in depth, in the context of the CoS.

2.3 Task Scheduling

In this section, we propose a formal definition of the task scheduling problem in CoS. We base our definition mainly on the definition of task scheduling for parallel programming presented by Sinnen [18].

Firstly, we define a task as a primitive and atomic (finest-grained, non-divisible) unit of execution of a program implemented by the VN. Thus, each task denotes a computing function in a VN program, capable of commanding a given physical node, such as the PSANs at the sensors tier. An example of a task is the collection of data (the reading operation performed by a sensing unit) regarding a given environmental variable at a moment in time. Another example of task is the processing of collected data to provide a more accurate or complete result on a given phenomenon.

The VN programs considered in task scheduling can have arbitrary task structures, with dependencies among the tasks of the same VN program. Thus, tasks have precedence relations and are dependent on each other (a data can only be processed

after it was collected). We represent a VN program as a Directed Acyclic Graph (DAG) G, described by Eq. (9).

$$G = (V, E, w, c) \tag{9}$$

In this DAG G, a vertex (in set V) reflects a task and a directed edge (in set E) is a precedence constraint between the incident nodes. Thus, the origin vertex must be executed before the destination vertex. Moreover, we associate weights (positive values) in w and c with the DAG nodes and edges, respectively. The weights in w and c represent the computation and communication costs of each task, respectively.

We consider that the DAGs from all the VN programs are connected to a unique DAG called UG. The objective of the UG is to provide a global view of all VN programs that run during the time window to which the scheduling occurs. The UG is created through a function f^s_{UG}, as shown in Eq. (10).

$$UG = f^s_{UG}(G_1, G_2, \ldots) = (UV, UE, Uw, Uc) \tag{10}$$

The function f^s_{UG} must implement the basic procedures required to build UG, but also offers the opportunity to implement other procedures to preprocess the DAGs, such as finding tasks in common among DAGs G, and merging such common tasks in UG, as shown in [31]. We describe the basic procedures to connect all VN programs and thus build the UG as follows. Besides the vertices and edges of the DAGs G, the UG has two additional vertices called start and end vertices, which are dummy vertices (having no costs). The start vertex of UG is connected through additional edges to the start vertices of each DAG G. Similarly, the end vertex is connected to the end vertices of each DAG G. Thus, formally, the challenge of task scheduling is to find spatial and temporal assignments of the vertices (UV) from UG onto the PSANs, which result in the best possible execution (evaluated through costs Uw and Uc), while respecting the precedence constraints expressed by the edges in UE.

It is important to mention that the spatial and temporal assignment resulting from task scheduling fundamentally determines the efficiency of the execution of the whole set of VN programs, which may share the same PSANs. This efficiency can be defined, for instance, in terms of reducing the costs (weights Uw and Uc), or several other types of costs, such as energy, or time used for performing the set of applications. Moreover, the most general scheduling problem imposes no restrictions on the DAGs; i.e., they may have arbitrary computation and communication costs as well as an arbitrary structure, and the number of PSANs is limited. Several other scheduling problems (special cases) arise by restricting the DAGs, for example, by having unit computation or communication costs, or by employing an unlimited number of PSANs. Thus, for the most general scheduling problem (and every special case), we formally define a spatial assignment function and a temporal assignment function.

As shown in Eq. (11), a spatial assignment S of the UG on a finite set S_PSAN of all PSANs is the function $f^s_{spatial}$ of the nodes of UG to the PSANs of S_PSAN.

$$S = f^{s}_{spatial} : UV \rightarrow S\_PSAN \tag{11}$$

As shown in Eq. (12), temporal assignment T of the UG on a finite set S_PSAN of all PSANs is the function $f^{s}_{temporal}$, which is the start time of the nodes of UG.

$$T = f^{s}_{temporal} : UV \rightarrow \mathbb{Q}^{+}_{0} \tag{12}$$

Therefore, a full schedule (FS), which is the output of the task scheduling process of a VN, refers to the pair of both S and T, as shown in Eq. (13).

$$FS = (S, T) \tag{13}$$

Finally, the task scheduling solution should be designed to make the best decisions, aiming to achieve two potentially conflicting objectives: (i) maximize the meeting of requirements posed by the tasks, while (ii) seeking the lowest possible consumption of PSANs resources. In several aspects, the process of task scheduling in CoS resembles the process of resource allocation in CoS. However, both processes differ mainly with respect to the nature (physical or virtual) of the resources to be allocated. Task scheduling must deal with the interface between physical and virtual devices, while resource allocation deals with the relations fully within the virtual realm (among VNs and software CoS applications). Moreover, the tasks are defined as atomic units run by physical nodes, therefore at a finer grained level than application requests. Finally, an analogy can be made between VN programs and software CoS applications. Section 2.4 provides further detail on this aspect, on software CoS applications and on the whole process of resource allocation.

2.4 *Resource Allocation*

In this section, we propose a formal definition of the resource allocation problem in CoS. This definition is similar to the definition of task scheduling presented in Sect. 2.3; thus, we focus on describing the differences of context from both definitions.

Resource allocation is a subject addressed in many computing areas, such as operating systems, grid computing, and data center management. As the authors in [32] defined for cloud computing, the problem of resource allocation aims to guarantee the correct meeting of application requirements by the physical infrastructure, abstracted through virtual machines, while minimizing the operational cost of the cloud environment. Resource allocation in CoS has a similar goal. It refers to the process of allocating the VN instances to perform the application requests (workload) submitted to the CoS by users, attempting to maximize the meeting of applications requirements, while respecting the constraints of VNs and, therefore, the constraints of the underlying physical infrastructure. It differs from resource allocation in cloud computing by dealing with the specificities of the three-tier CoS physical infrastruc-

ture (comprising the sensor, edge, and cloud tiers). In the CoS, the main resource provided by VNs to applications is the data obtained from the physical infrastructure, in contrast to computational, storage, and communication capabilities in traditional cloud computing. Therefore, the resource allocation solution should be designed to make the best decisions, aiming to achieve two potentially conflicting objectives: (i) maximize the meeting of requirements posed by the application requests, while (ii) seeking the lowest possible consumption of resources exposed through VNs.

Regarding objective (i), the resource allocation process should take into account, mainly, the requirements of data freshness, data processing complexity, and response time required by the application [33, 34]. Applications demanding fresher data than currently stored in a virtual node will require the direct engagement of the physical nodes. On the other hand, applications wishing information resulting from complex processing, often involving the use of historical data and trends or the result of calculation of time series, are preferably met by the cloud. Finally, applications that demand the detection of events involving one or more types of sensing data, and that need immediate response when these events occur, usually do not tolerate the high latencies resulting from sending data to the cloud and further processing. In this case, performing the processing at the edge tier is the best option.

Regarding objective (ii), considering that in the CoS there are many requests arriving, there are applications that can share outputs of the current run (or previous runs) of a given service in a virtual node. By leveraging such sharing, it is possible to avoid that the service runs every time a new request arises, thus saving CoS infrastructure resources. Any existing physical node in one of the CoS tiers (sensors, edge, and cloud tiers) can store and make available the data related to the execution of the services. This physical node may belong to the virtual node that performed the service in question, or eventually not, belonging to the set of physical nodes subordinate to another virtual node. Thus, certain future requests may never demand the execution (in fact, in physical nodes) of a service, relieving the respective load of tasks in physical nodes. This decision regarding the resource allocation in CoS [35] should aim for solutions in which the data are provided for applications with minimal costs, but without harming the requirements of data freshness and response time.

Regarding a formal definition of the resource allocation problem in CoS, firstly, we consider that nodes from graph G in Eq. (9) comprise requests that form an application, instead of tasks that form a VN program. Next, in Eq. (14) we consider a function f^r_{UG}, which forms the Unique DAG of requests for all applications, with similar rules to the ones described for task scheduling in Sect. 2.3.

$$UG = f^r_{UG}(G_1, G_2, \ldots) = (UV, UE, Uw, Uc) \tag{14}$$

Formally, the challenge of resource allocation is to find spatial and temporal assignments of the vertices (UV) from UG onto the VNs, which result in the best possible execution (evaluated through costs Uw and Uc), while respecting the precedence constraints expressed by the edges in UE. For the most general resource allocation problem (and every special case), we formally define a spatial assignment

function and a temporal assignment function. As shown in Eq. (15), a spatial assignment A of the UG on a finite set S_VN of all VNs is the function $f^r_{spatial}$ of the nodes of UG to the VNs of S_VN.

$$A = f^r_{spatial} : UV \rightarrow S\_VN \tag{15}$$

As shown in Eq. (16), temporal assignment T of the UG on a finite set S_VN of all VNs is the function $f^r_{temporal}$, which is the start time of the nodes of UG.

$$T = f^r_{temporal} : UV \rightarrow \mathbb{Q}^+_0 \tag{16}$$

Therefore, a full allocation (FA), which is the output of the resource allocation process of a VN, refers to the pair of both A and T, as shown in Eq. (17).

$$FA = (A, T) \tag{17}$$

Finally, considering all the steps involved in the task scheduling and in the resource allocation processes for CoS, solutions are required to tackle these challenges. Section 3 reviews the state of the art in such solutions.

3 State of the Art

In subsections (Sects. 3.2 and 3.3), we review the state of the art on solutions for task scheduling and resource allocation, respectively, highlighting how each solution approaches the challenges involved in such activities. Initially (Sect. 3.1), we describe the criteria to be used throughout the text to organize the solutions that will be presented. As task scheduling and resource allocation share several characteristics and objectives, the same criteria are used to analyze the proposals for both activities. We conclude with Sect. 3.4, in which we present a summary of the results of our review from the state of the art in task scheduling and resource allocation applied to CoS.

The concepts of resource allocation and task scheduling are not new in the literature. Several areas of research closely correlated to CoS, such as IoT and WSANs, study these concepts with slight differences of context. Such differences relate mainly to the nature of the resources to be scheduled or allocated. In the following sections, we chose, mainly, to depict the works that pertain to our area of research, the CoS. However, since the area of CoS is recent, only few studies exist proposing task scheduling and resource allocation solutions. Therefore, we will discuss task scheduling and resource allocation solutions under a broader perspective, including the areas of IoT and WSAN. We choose the most recent solutions from other areas than the CoS that require little or no adaptation when applied to operate in the CoS.

It is important to mention that several resource allocation and task scheduling solutions exist for the area of cloud computing [36], which is also closely related to

CoS. Such solutions ignore the specificities of the CoS environment. In traditional cloud computing, the physical infrastructures contain several data centers with high computing capacity servers, where the most important is simply to provision their processing capabilities, storage, and communication data. When operating in a CoS environment, the need for provisioning sensing and actuation, the heterogeneity and the computational resource constraint nature of devices pose new challenges to perform the resource provisioning. However, the area of resource allocation and task scheduling for cloud computing has already been extensively surveyed, and thus, we excluded this area from our review of the state of the art.

Finally, based on the background provided in Sect. 2, and that the best theoretical position for positioning VNs is the edge tier [8], we assume that task scheduling refers to the relation between the sensors tier and the VNs, while resource allocation refers to the relation between VNs and the edge/cloud tiers of the CoS. All the reviewed proposals are developed considering one of these relations. Thus, our first macro-criterion to classify proposals is to divide them into "proposals for task scheduling that can be applied at the sensors tier" and "proposals for resource allocation that can be applied at cloud and edge tiers." Within these two macro-categories, we identify and classify the existing proposals based on a set of additional criteria, described in the following subsection.

3.1 Classification Criteria

As a **first criterion** for classifying proposals for task scheduling and resource allocation, we consider the decentralization degree of solutions. In general, although centralized solutions are simpler to implement and have a global view of the network, they are well known as being more susceptible to failures and less scalable. In centralized task scheduling algorithms, a single VN placed at the sink node in the WSAN or a manager node in the cloud is responsible for deciding how, when, and where to perform the tasks. In turn, in decentralized task scheduling algorithms, multiple PSANs, known as the local schedulers, determine the scheduling. The local schedulers do not need to perform scheduling for all nodes, because different nodes may have different areas of interest, contributing to the scalability of the solution. Kaur Kapoor et al. [37] show that, among the several centralized and decentralized algorithms they proposed, the performance of the best-centralized algorithm is comparable to that of the best-decentralized algorithm for smaller systems. For medium and high system loads, the decentralized algorithms demonstrate a significantly higher performance in comparison to the centralized algorithms. Moreover, a hybrid approach can be adopted (partially decentralized), sharing characteristics of both centralized or decentralized solutions, sometimes combining two decision phases, one centralized and the other decentralized.

As a **second criterion** to classify existing solutions, we consider the ability of considering priorities during the task scheduling or the resource allocation process. Priorities can be considered among different application requests, tasks of VN pro-

grams, VNs, or PSANs. In the context of resource allocation, there may be applications requests with higher priority, concerning their degree of criticality (response time). For instance, a HVAC application may be less critical than a fire detection application. This priority can also concern the amount of resources provided to the applications (VNs and PSANs), compared with other applications that are sharing the physical infrastructure. Regardless the kind of priority, such information regarding priority, possibly provided by users, could be used to rank tasks, requests, VNs, and the PSANs, which should be assigned first during task scheduling or resource allocation. For instance, in resource allocation, if a more priority application arrives, the VN must stop the request currently running and queue the requests of the highest priority application according to its priority.

As a **third criterion** to be used when analyzing and classifying existing proposals, we consider their ability to handle precedence relationships (dependencies between inputs and outputs) among requests and among tasks. Most works, such as [12], consider applications/VN programs that comprise a single request/task regarding data acquisition. This is a common approach, for instance, in task scheduling algorithms for traditional WSANs. In such algorithms, the VN programs have the interest on mere data acquisition (represented as a single data acquisition task), without considering the actuation, computation, and communication capabilities of WSAN nodes. In traditional WSANs, the task scheduling algorithm decides only about which nodes will perform each single task, and at which time such execution will occur. Thus, since tasks have no precedence relationships, there is no need to model VN programs through, for instance, the usual representation of a DAG mentioned in Sect. 2.3. However, in scenarios where VN programs consider multiple tasks (possibly mixing data acquisition, actuation, computation, and communication tasks) with precedence relationships, it is very important to model such relationships, in order to make a proper decision. This applies to either task scheduling or resource allocation, because PSANs or VNs must perform first the requests or tasks whose inputs are satisfied and are free to start being processed. It is important to mention that the representation of requests and tasks through a DAG and handling the precedencies among them is a feature explored by few works. Moreover, it is one first step toward another feature not explored by any work so far, which is representing an application or VN program as a complex structure of standard information fusion procedures (each request or task as being an information fusion procedure).

We consider the ability of sharing the results of execution among requests or tasks as a **fourth criterion**, which is relevant to analyze the proposals and their benefits. Some solutions for task scheduling share the results of tasks that are common among multiple VN programs. These proposals perform the tasks in common (i.e., tasks that will serve different VN programs) only once, sharing the results to further improve the resource utilization of PSANs.

As a **fifth criterion** for classifying proposals, we consider the adoption of what we call a *full device virtualization model* by these proposals. A full virtualization model is defined as a process to virtualize the data and computational/communication capabilities from PSANs through the instantiation of VNs, also comprising task

scheduling and resource allocation procedures (mostly adopted by the proposals for the areas of CoS and IoT).

As a **sixth criterion**, we consider the ability of supporting time-based (pull) and event-based (push) applications (or VN programs) simultaneously. A time-based (TB) application is the one for which the application decides the exact moment in time for demanding the VN (or PSANs) resources. Thus, the pull model attends time-based applications (or VN programs). For instance, in the context of a pull model in resource allocation, a VN demands its underlying physical infrastructure and responds to the application at the time defined by the application. This is the most conventional type of application, and it is perfectly suited to the context of resource allocation due to the temporal aspect; i.e, VNs can have full control about the time duration of each request, and do not need to wait for events with random occurrence times in future (interrupts), to finish their processing. Thus, pull applications (VN programs) are supported by all resource allocation (task scheduling) proposals to be discussed in Sects. 3.2 and 3.3. An event-based (EB) application (VN program) is the one that demands the VN (PSAN) resources only when it detects the occurrence of a specific event of interest by the application (VN program). Thus, a push model attends event-based applications (VN programs). For instance, in the context of a push model in resource allocation, the VN demands its resources at a moment in time that is unknown, previously to the occurrence of the event of interest by the application. The push model is a typical publish/subscribe model, in which applications subscribe their interests to a VN, which, in turn, waits until an event occurs to publish its results respective to this event. The ability to support both models simultaneously makes the approach for resource allocation or task scheduling more general, thus fitting well in a high-scale deployment of a CoS infrastructure, meeting a broader range of applications.

As a **seventh criterion**, we classify proposals according to the characteristics of the optimization problem formulated to meet the resource allocation and task scheduling objectives. There are several methods in the literature that can be used to mathematically formulate the resource allocation and task scheduling problems [38, 39]. In most cases, the optimization problem is referred as an integer program, in which the decision variable (with respect, for instance, to which PSAN will be allocated to which task) is a binary variable. Some other real-valued criteria may also be of use, and thus could be included in the problem formulation, such as when defining the start times of tasks. When including real-valued decision variables, the optimization problem is referred as a linear program. Another commonly used option is to formulate the problem as a multi-objective optimization problem [39]. In contrast to the previous cases of single-goal problems, the multiple goals conflict with each other, i.e., to maximize the attendance to application requirements results in an increased consumption of resources, and vice versa. That is, it is not possible to find a single solution that minimizes an objective and maximizes another simultaneously. In the multi-objective approach, it is obtained an optimal solution set (Pareto-optimal solutions) with numerous solutions indifferent to each other, according to some pre-established criteria, leaving the analyst to decide which solution to use, according to its own established criteria.

As an **eighth criterion**, we classify proposals according to the characteristics of the algorithm/heuristic used to solve the formulated optimization problem. To solve optimization problems in order to seek the optimal solution, there are a number of methods in the literature [39, 40]. One example is the linear programming and its variants [40]. However, when considering the typical approach of representing requests and tasks as DAGs, the optimization problem becomes more difficult as the size of the DAG increases. In fact, finding a schedule of minimal length for a given DAG is, in its general form, an NP-hard problem [18, 38], i.e. problems whose explosive combinatorial nature hinder the quick search for optimal solutions when they grow [41]. That is, an optimal solution cannot be found in polynomial time (unless NP = P). Because of this kind of problem, an entire area emerged that deals with quick search of solutions, ranging from the theoretical analysis to heuristics and approximation techniques that produce near-optimal solutions. Thus, the problems tackled in resource allocation and task scheduling fit the class of NP-hard problems, justifying the application of heuristic techniques.

Heuristic techniques approach the optimal solution, solving the problem by obtaining sub-optimal solutions in reduced computation time, once the search for the optimal solution is much more computationally intensive. Heuristic-based proposals can follow, for instance, traditional graph theory, evolutionary, game/auction, greedy, machine learning, Voronoi diagrams, or probabilistic approaches [36]. Evolutionary algorithms are typically used to provide good approximate solutions to problems that cannot be easily solved using other techniques. Due to its random nature, it is not guaranteed that evolutionary algorithms find an optimal solution to the problem, but they will often find a good solution, if any. Genetic algorithms have proven to be a successful way to produce satisfactory solutions to many problem formulations. Finally, game theory is increasingly being used as a modeling and design framework in decentralized algorithms. A distributed game-theoretic approach to task allocation provides autonomy to sensor nodes, which can decide the best scheduling in actual neighboring context. In solutions following such approach, communications are made only between certain sensors in a neighborhood, which is a potentially energy-efficient and scalable solution.

As a **ninth criterion**, we consider the presence of the edge tier in the proposed architectures [42]. In such proposals, the devices in the edge tier (edge nodes, sinks, gateways, for instance) play an active role in the resource allocation or task scheduling process, allowing solving the problem in a distributed way.

Finally, the **tenth criterion** regards considering physical devices (sensors/ things) as active resource providers, providing to applications (VN programs) computing and actuation capabilities, and not being mere passive sensing data sources.

3.2 State of the Art on Task Scheduling at the Sensors Tier

In this section, we describe relevant works found in literature that represent the state of the art on task scheduling at the sensors tier of the CoS architecture. The described

works present proposals either tailored for the specific field of CoS [42–45], or for the broader field of IoT [46–50], or for the more traditional field of WSAN [12, 28, 31, 37, 51–57].

Among the works that propose task scheduling for CoS, Zhu et al. [43] is an example of a centralized solution. In their work, the authors analyze the characteristics of task scheduling with respect to integrating cloud computing and WSANs, proposing two novel task scheduling algorithms. Their algorithms follow a heuristic, greedy-based approach and are able to handle priorities among tasks. In their algorithms, the main goal is to divide tasks into two groups: tasks to be performed in the WSAN (G1) and tasks to be performed in cloud (G2). For all tasks submitted in group G1, with higher priority, the algorithm uses a set of rules to decide (based on the costs of using the resources available) for the best schedule. After, the same procedure is performed for group G2, with lower priority.

Phan et al. [42] is another example of a centralized solution. They proposed a cloud-integrated WSAN architecture and studied the optimization of a push-pull communication scheme among the three layers of their architecture using a genetic algorithm. Therefore, they are among the proposals that can support both time-based and event-based applications, which guarantees they cover a broad spectrum of application domains. In addition, in this proposal the devices in the edge tier (edge nodes, sinks, gateways, for instance) play an active role in the task scheduling process. Moreover, their proposal includes a full device virtualization model. At the sensor layer, several heterogeneous WSANs embedded in the physical environment exist, using a tree topology. Nodes periodically read sensors and push data to the sink node. The edge layer is a collection of sink nodes, each of which participates in a certain sensor network and stores incoming sensor data in its memory, pushing them periodically to the cloud layer. Sink nodes maintain the mappings between physical sensors and virtual sensors. In addition, each sink receives a "pull" request from a virtual sensor when the virtual sensor does not have the data that an application requires. If the sink node has the requested data in its memory, it returns that data. Otherwise, it issues a pull request to a sensor node that is responsible for the requested data. The cloud layer operates on one or more clouds to host end-user applications and management services for the applications. Applications are operated on virtual machines in clouds, and always access physical sensors through virtual sensors. Users are assumed to place continuous sensor data queries on virtual sensors via cloud applications in order to monitor the physical environment. If a virtual sensor already has data that an application queries, it returns that data. If a query does not match, the virtual sensor issues a pull request and sends it to a sink node. Phan et al. focus on two services in their virtualization model. The first is the sensor manager, which virtualizes physical heterogeneous sensors in a unified way by abstracting away their low-level operational details. The second is the communication manager, responsible for push-pull hybrid communication between different layers. The key component in the communication manager is the communication optimizer, which solves an optimization problem to seek the optimal data transmission rates for sensor and sink nodes with respect to multiple optimization objectives (maximize sensor

data yield for applications, minimize bandwidth consumption between the cloud and edge layers and minimize energy consumption in the sensors layer).

Dalvi [44] proposed a centralized task scheduling scheme to minimize energy consumption in CoS, which is based on TDMA, spatial correlation, and on the Voronoi diagram. The Voronoi diagram is used for representing WSANs in a CoS. In the diagram, a rectangular region represents the area from which the user needs data. The area covered by the WSAN is divided into small cells that are centered at points. Each point in the cell represents a wireless sensor in a WSAN. The edges of each cell are formed by connecting perpendicular bisectors of the segments joining all neighboring points. The sensor located at the center of a cell can sense data for the area covered by cell. This data is more accurate as compared to the data sensed by other sensors for that region. This Voronoi diagram is built when WSN is initialized. A calculation is performed to find the minimum number of nodes required to cover the area selected by the user. The allocation scheme is based on the concept that sensors in densely deployed zones will have more number of neighbors compared to sparsely deployed zones and hence more number of edges in Voronoi diagram. As the previously discussed work [42], the solution presented in [44] also supports both pull and push communication models and includes a full device virtualization model, similar to the one in [42]. In the virtualization model proposed by Dalvi, there are three layers: client centric, middleware, and sensor centric. The client centric layer connects end users to the CoS, managing a user's membership, session, and GUI. Middleware is the heart of the CoS, managing virtual sensors with help from components such as provision management, image life cycle management, and billing management. The sensor centric layer connects the middleware to the physical WSNs. It also registers participating WSNs, maintains participating WSNs, and collects data.

Yao et al. [45] proposed a centralized and adaptive task scheduling mechanism to schedule optimally the transmission opportunities of devices, considering multimedia distortion reduction, hidden node problem, transmission interference, and signal coverage. Their proposed mechanism adopts a heuristic-based, greedy approach for performing the task scheduling.

Among the works that propose task scheduling solutions for IoT, Kim and Ko [47] present a centralized task scheduling approach based on genetic algorithm to minimize the amount of data transmissions between mobile devices in IoT. They transformed the task scheduling problem into a variant of the degree-constrained minimum spanning tree problem and applied a genetic algorithm to reduce the time needed to produce a near-optimal solution.

Li et al. [46, 49] and Billet et al. [50] are also examples of centralized solutions for task scheduling in IoT. Li et al. [46] proposed a genetic algorithm based on a teaching and learning technique for scheduling tasks with multiple restrictions of relations in processing sequences in IoT. In their genetic algorithm specification, the crossover operator is used in the swarm intelligent algorithms to learn information from other solutions, thus converging to the optimal search space faster. Li et al. [49] proposed a QoS scheduling model for IoT, which explores optimal QoS-aware services composition at application layer, heterogeneous network environment at network layer,

and the information acquisition for different services at sensing layer. Billet et al. [50] proposed a binary programming problem formulation for task scheduling for IoT, along with an efficient heuristic for solving it, based on location, capabilities, and QoS constraints.

Among the works that propose task scheduling solutions for WSANs, de Farias et al. [31] proposed a framework for Shared Sensor and Actuator Networks (SSAN), including an energy-efficient centralized task scheduling algorithm. A major feature of their work is that the algorithm performs tasks in common to multiple applications only once. In other research, Li et al. [28] introduce a task scheduling algorithm exploiting the fact that different applications may share the same sensing data with common QoS requirements, as well as spatial and temporal characteristics. Both proposals promote the cost-effective utilization of the resources available in the shared infrastructure, aiming to increase the return of the investment (ROI) for the owners. They support different priorities and precedence relationships among tasks. Furthermore, both support time-based (pull) and event-based (push) applications simultaneously. Also in the context of SSANs, Bhattacharya et al. [56] proposed an integrated application deployment system that performs task scheduling based on their Quality of Monitoring (QoM) of physical phenomena due to the close coupling of the cyber and physical aspects of distributed sensing applications. Therefore, the task scheduling algorithm deals with the inter-node Quality of Monitoring (QoM) dependencies, typical in cyber-physical applications. The QoM of a distributed sensing application usually depends on the set of nodes allocated to it. Moreover, the measurements of different sensors are often highly correlated resulting in inter-node dependency; i.e., the QoM contributed by a node to an application is dependent on the other nodes allocated to the same application. Since the SSN paradigm aims at fully exploiting the deployed sensing infrastructure for executing multiple applications with distinct requirements, it is a desirable feature that all possible communication patterns are support. Therefore, the work described in [56] schedules both time-based and event-based applications. However, the authors do not mention the handling of different priorities, which is another desirable feature for SSNs.

Our research group, in [28, 31], proposed centralized algorithms for scheduling tasks in WSANs. These algorithms, in addition to being concerned about saving energy by choosing the best node for a particular task, also perform tasks in common for different applications only once, sharing the results of these applications to improve the use of the limited resources of physical nodes. This sharing takes advantage of the fact that the same physical infrastructure is used by multiple applications, as is the case in virtualized environments. These tasks in common are identified by preprocessing the DAGs of applications, before starting the task scheduling process. These two works also include a full device virtualization model, in which the sensors play an important role as service providers. In their virtualization model, three major elements exist, namely Web server, sink nodes, and sensor nodes, organized in a hierarchical manner. The Web server acts as a frontier to handle arrivals of applications and performing task scheduling. Final users, through applications, request different services from the system and the Web server acts as a service provider to reply those requests. After receiving requests from the Web server, the sink node of each WSN

schedules these tasks to individual sensor nodes. Sensor nodes send the descriptions of their services to sink nodes, which keep them in a repository.

Dai et al. [51] propose a multi-objective algorithm for centralized task scheduling in WSANs, seeking to optimize the total make span of tasks, but meanwhile, also paying attention to the probability of node failure (related to unsuccessful task performing) and the lifetime of network.

Hu et al. [54] propose three different greedy-based centralized algorithms to optimize data fusion parameters in the WSAN task scheduling problem, modeled as an Integer Linear programming problem. Their proposal includes a full device virtualization model. In their model, a given virtual sensor is instantiated and cannot change during a given time slot. They assume a common sensing period, with each sensor generating samples and making a local decision as to whether an interesting event occurred. Event arrival times are independent, and their distribution is known in advance. The events of interest are, in theory, detectable by the available physical sensors. Their virtualization model comprises two pre-deployment-independent functions: a function for forming information fusion rules, and a function for assigning optimal virtual sensors and their scheduling. Fusion rules are formulated through either synthetic or experimental data. The intuition is that the fusion rules define the optimal set of sensors for each event as well as the algorithm to combine the sensor readings.

Rowaihy et al. [12] proposed centralized and decentralized energy-aware solutions to the problem of optimally scheduling multiple missions to WSANs, in which each mission uses its specific and exclusive subset of sensor nodes. The problem of mission-sensor scheduling is modeled as a weighted bipartite graph to optimally schedule the sensors for missions. Their proposed solutions build on traditional graph theory and are able to handle priorities among missions, relating priority with mission profit; i.e., these solutions schedule missions that have higher profit first than other missions (missions are sorted in order of decreasing profit). In the proposed weighted bipartite graph, the vertex sets consist of sensors and tasks. A positively weighted edge means that a sensor is applicable to a task. The weight of the edge indicates the utility that the sensor could contribute to the task. The authors seek a semi-matching of sensors to tasks, so that (ideally) each task is satisfied. Their proposed solutions perform graph manipulation to represent different problem variants, with different constraints. Moreover, they propose several greedy algorithms, jointly with the graph theory approach. One of their greedy algorithms considers tasks in decreasing order of profit. For each task, the algorithm assigns available sensors in decreasing order of offered utility, until the mission is satisfied.

Li et al. [57] and Edalat et al. [55] are examples of hybrid approaches, meaning that they inherit features of both centralized and decentralized solutions. In Li et al. [57], the authors proposed a heuristic-based three-phase algorithm for allocating tasks to multiple clusters in hierarchical WSANs, seeking to minimize the overall energy consumption and balancing the workload of the system while meeting the applications deadlines. The task scheduling problem is referred as an integer program and solved as a multi-objective problem. The proposal considers tasks with different priorities and dependencies among the data input and outputs of tasks, represented

through DAGs. Moreover, the sensors are considered as active resource providers, instead of being mere passive data sources. Edalat et al. [55] proposed a heuristic two-phase winner determination protocol to solve the task scheduling problem modeled as a distributed reverse combinatorial auction, seeking to maximize WSAN lifetime while enhancing the overall application QoS, in terms of deadlines. They also consider tasks with different priorities and dependencies among the data input and outputs of tasks, represented through DAGs.

In terms of distributed solutions, Wu et al. [52] present a distributed game-theoretic approach for task scheduling in SSANs based on the correlation among sensor readings from different nodes. Their approach supports time-based and event-based communications.

Wang et al. [53] propose a cluster-based task scheduling algorithm based on a greedy approach that balances the node energy consumptions, where the sensor nodes are classified into different ranking domains and that supports requirements of real-time, heterogeneity, flexibility, and scalability. Different priorities among application tasks are supported and the precedence relationships of tasks are considered in their scheduling decisions.

Kim [48] proposed a task scheduling solution to optimize bandwidth in IoT based on a cooperative game and on the concept of Shapley value. They adopt a fully decentralized approach and leverage the collaboration among the devices to achieve the application goals. Their approach is able to handle tasks with different priorities.

Finally, besides comparing proposals for task scheduling regarding their decentralization degree, Kaur Kapoor et al. [37] proposed their own greedy-based decentralized task scheduling algorithms. They proposed the random allocation algorithm, which does not use any information about the application or the network while making a scheduling decision. They proposed the CPU Load Balanced Allocation and Data Load Balanced Allocation algorithms, which schedule tasks making decisions to balance the CPU usage and data transmissions, respectively, among nodes. They also proposed the Balanced Metric Allocation algorithm, which aims to balance the energy consumption, both due to the CPU component and the radio component among all the sensor nodes. Finally, they also proposed the Maximum Energy First algorithm, which selects first, from the set of available nodes, the sensor nodes that have the highest available energy, for execution of the tasks.

Table 1 presents a summary with the task scheduling proposals classified according to the 10 criteria considered in this chapter. In the following section, we present the state of the art on resource allocation at edge and cloud tiers, and we classify the respective proposals according to the same 10 criteria used in this section.

Table 1 Classification of task scheduling proposals

	1 Degree of decentralization			2 Handles priorities	3 Handles precedencies	4 Shares requests/tasks in common	5 Full device virtualization model	6 Type of application		7 Optimization problem formulation			
	1.1 Centralized	1.2 Hybrid	1.3 Decentralized					6.1 Time based	6.2 Event based	7.1 Integer	7.2 Linear	7.3 Nonlinear	7.4 Multi-objective
[56]	X							X	X	X			
[31]	X				X	X	X	X	X				
[28]	X			X	X	X	X	X	X				X
[49]	X			X			X	X			X		
[46]	X				X			X					
[50]	X				X			X	X	X	X		
[45]	X							X					
[43]	X			X				X					
[54]	X						X	X		X			
[42]	X						X	X	X				X
[51]	X							X					X
[44]	X						X	X	X				
[47]	X							X					
[55]		X		X	X	X		X	X	X			
[57]		X		X	X			X		X	X		X
[12]			X	X				X		X	X		
[52]			X					X	X	X			
[37]			X					X					
[53]			X	X	X			X					
[48]			X	X				X	X				

(continued)

Table 1 (continued)

	8 Approach of the algorithm proposed/used to solve the problem								9 Considers the edge tier	10 Devices as active resource providers
	8.1 Graph theory	8.2 Evolutionary	8.3 Game /auction	8.4 Greedy	8.5 Machine learning	8.6 Voronoi Diagram	8.7 Lin. programming	8.8 Probabilistic		
[56]				X						
[31]	X			X						
[28]	X			X						X
[49]				X						
[46]		X			X					
[50]				X						
[45]				X						
[43]				X						
[54]				X						
[42]		X							X	
[51]		X								
[44]						X				
[47]	X	X								
[55]			X							
[57]	X			X						X
[12]	X			X						
[52]			X							
[37]				X						
[53]				X						
[48]			X							

3.3 State of the Art on Resource Allocation at Edge and Cloud Tiers

In this section, we describe relevant works found in literature that represent the state of the art on resource allocation at the edge and cloud tiers of the CoS architecture. The described proposals were developed either for the field of CoS [15, 16, 58, 59] or IoT [60–67]. We will analyze the proposals in the light of the same criteria used for task scheduling solutions. Almost all works, with a single exception, propose centralized solutions for the resource allocation problem.

Among the works that propose resource allocation in CoS, Delgado et al. [15, 16] proposed an heuristic algorithm and optimization framework to perform resource allocation, seeking to maximize the number of applications sharing the CoS, while accounting for the limited storage, processing power, bandwidth, and energy consumption requirements of sensors tier. Their resource allocation algorithm follows a traditional linear programming technique.

Misra et al. [58] present a mathematical formulation of CoS, with a thorough evaluation of the cost effectiveness of CoS by examining the costs of sensor nodes due to deployment, maintenance, and rent by users, as well as the profits in terms of the service acquired from the sensed data, always from the perspective of every user of the CoS. In the full device virtualization model used by Misra et al., there are three tiers (users/applications, virtual nodes, and sensors). The communication interface of a user is primarily a Web interface running at the site of the cloud service provider. It is a Web portal, through which the user requests the CoS. The user is kept abstracted from the underlying complex processing logic required due to perform resource allocation, application-specific aggregation, and virtualization. Moreover, sensor nodes are heterogeneous; thus, the sensor nodes are standardized using a Sensor Modeling Language. Every physical sensor node reports its sensed data to the CoS storage. Within the cloud environment, the sensed data are aggregated in real time.

Dinh et al. [59] propose an interactive model for the CoS to provide on-demand sensing services for multiple applications with different requirements, designed for both the cloud and sensor nodes to optimize the resource consumption of physical sensors, as well as the bandwidth consumption of sensing traffic. Dinh et al. consider requests in common in their solution. Their approach formulates a unique DAG of requests, merging requests in common for multiple applications. In their approach, requests in common are merged by considering the more restrictive application requirements.

Among the works that propose resource allocation in IoT, Narman et al. [60] propose a dedicated server allocation for heterogeneous and homogeneous systems, to provide efficiently the desired services by considering priorities of applications requests. Yu et al. [61] proposed a cloud-based vehicular network managed by a strategy based on game theory to optimally allocate resources, together with virtual machine migration. Both proposals include a full device virtualization model. The first work [60] also includes mechanisms to handle priorities among application

requests, using this priority value to decide the next request to be served and allocate the amount of resources for each request.

Angelakis et al. [63] presented a mathematical formulation of assigning services to interfaces with heterogeneous resources in one or more rounds and developed two algorithms to approximate the optimal solution for big instance sizes.

Zeng et al. [62] proposed an edge computing supported software-defined embedded system, together with the formulation of a resource allocation problem as a mixed-integer nonlinear programming problem, and a computation-efficient solution. Vögler et al. [64] propose an infrastructure that provides elastic provisioning of application components on resource-constrained and heterogeneous edge devices in large-scale IoT deployments, which supports push-based as well as pull-based deployments. Moreover, their infrastructure also manages time precedence restrictions among application tasks. Aazam et al. [65], in their proposed methodology for resource estimation and management through edge computing, formulate resource management based on the fluctuating relinquish probability of the customer, service type, service price, and variance of the relinquish probability. In their paper, they extended their previous model to include a customer probabilistic resource estimation model, to manage the resources for IoT devices. Different priorities among application tasks are considered in their solution. Abedin et al. [67] provide an efficient IoT node pairing scheme between the same domains of IoT nodes in edge paradigm, based on the Irving's matching algorithm, and model the problem as a one-sided stable matching game with quota. All these works [62, 64, 65, 67] consider the presence of the edge tier in their proposed architectures.

Aazam et al. [66] present a service-oriented resource management model for IoT devices, using edge computing. Their work is mainly focused on considering different types of services and providing device-based resource estimation and pricing, even in presence of mobility. In their proposed model, sensors, IoT nodes, devices, and cloud service customers (CSCs) contact the edge to acquire the required service(s) at best price. CSCs perform the negotiation and service-level agreement (SLA) tasks with the edge. The edge is in charge of estimating the consumption of resources, so that they can be allocated in advance.

Only the work described in [61] proposed a decentralized approach to resource allocation. The authors proposed to integrate cloud computing into vehicular networks such that the vehicles can share computation resources, storage resources, and bandwidth resources. Their proposal includes a full device virtualization model, including a vehicular cloud, a roadside cloud, and a central cloud. The authors study resource allocation and virtual machine migration for effective resource management in this cloud-based vehicular network. A game-theoretical approach is presented to optimally allocate cloud resources. Virtual machine migration due to vehicle mobility is solved based on a resource reservation scheme.

Table 2 presents a summary with the resource allocation proposals classified according to the 10 criteria considered in this chapter. In the following section, we present a summary of both the state of the art on resource allocation and task scheduling. We present a macro-classification of all works reviewed in this chapter, under the same 10 criteria described in Sect. 3.1, and highlight the open issues in the

state of the art in task scheduling and resource allocation applied to CoS based on this classification.

3.4 Open Issues

In this section, we present a summary of the results of our review from the state of the art on task scheduling and resource allocation applied to CoS. We categorized all the relevant proposals found in literature by their key differentials. We present this categorization in Table 3.

Regarding task scheduling, we can notice some open issues in existing works. First, despite the inherently distributed nature of CoS, IoT, and WSN, more than half of the proposals for task scheduling are centralized. Although decentralized scheduling algorithms have been proposed, they still lack an important feature that is to promote the sharing of common tasks among the VN programs. We believe this is a key requirement to achieve an efficient solution mainly in large-scale deployments. In addition, besides sharing the tasks of VN programs, we claim that efficient and suitable solutions for CoS must take into account the priorities and precedencies among tasks, representing tasks and their precedencies through a DAG. Other relevant issue that is still poorly exploited in existing works is the proposal of a full device virtualization model, considering devices as active resource providers, other than passive data sources.

Regarding proposals for resource allocation, only one work found in the current literature proposed a decentralized solution, while all the others are centralized approaches. Moreover, only one proposal [59] was found on resource allocation for CoS that shares the results of tasks in common among multiple applications. We identify only one proposal that considers time precedence restrictions among applications' tasks [64], which is also the only one supporting time-based (pull) and event-based (push) applications simultaneously [64]. Three proposals [60, 65, 66] manage priorities among applications tasks, so this is also an underexploited feature in resource allocation. We believe this issue requires further investigations; since in a CoS scenario of large dimension and serving multiple applications, it is crucial to assure that more critical applications receive their required resources with some priority. As expected, a fair number of proposals consider the edge tier in their architectures [62, 64, 65, 67]. Based on all the benefits such tier can bring, we believe this will be a trend, mainly for time critical applications and for scenarios with mobile devices. Finally, five proposals [63–67] consider the physical devices as actively engaging as the resource providers, instead of being mere passive data sources. We claim that fully utilizing the resources provided by the sensors tier is the best strategy to build cost-effective CoS systems. However, this feature is still poorly exploited in the proposals.

Regarding all task scheduling and resource allocation proposals, it is important to mention that most solutions are either fully static, or handle partially the dynamic characteristic of the CoS environment when running the scheduling decision in

Table 2 Classification of resource allocation proposals

	1 Degree of decentralization			2 Handles priorities	3 Handles precedencies	4 Shares requests/tasks in common	5 Full device virtualization model	6 Type of application		7 Optimization problem formulation			
	1.1 Centralized	1.2 Hybrid	1.3 Decentralized					6.1 Time based	6.2 Event based	7.1 Integer	7.2 Linear	7.3 Nonlinear	7.4 Multi-objective
[60]	X			X			X	X					
[63]	X						X	X		X	X		
[15]	X							X		X	X		
[16]	X							X		X	X		
[58]	X						X	X					
[64]	X				X		X	X	X				
[62]	X							X		X		X	
[65]	X			X				X					
[66]	X			X				X					
[67]	X							X					
[59]	X					X	X	X			X		
[61]							X	X					

(continued)

Table 2 (continued)

	8 Approach of the algorithm proposed/used to solve the problem								9 Considers the edge tier	10 Devices as active resource providers
	8.1 Graph theory	8.2 Evolutionary	8.3 Game /auction	8.4 Greedy	8.5 Machine learning	8.6 Voronoi Diagram	8.7 Lin. programming	8.8 Probabilistic		
[60]				X						
[63]				X						X
[15]							X			
[16]							X			
[58]										
[64]									X	X
[62]							X		X	
[65]								X	X	X
[66]								X		X
[67]			X						X	X
[59]							X			
[61]			X							

Table 3 Criteria for classifying proposals

Criteria		Works on task scheduling	Works on resource allocation
1 Degree of decentralization	1.1 Centralized	[28, 31, 42–47, 49, 50, 51, 54, 56]	[15, 16, 58–60, 62–67]
	1.2 Hybrid	[55, 57]	
	1.3 Decentralized	[12, 37, 48, 52, 53]	[61]
2 Handles priorities		[12, 28, 43, 48, 49, 53, 55, 57]	[60, 65, 66]
3 Handles precedencies		[28, 31, 46, 50, 53, 55, 57]	[64]
4 Shares requests/tasks in common		[28, 31, 55]	[59]
5 Full device virtualization model		[28, 31, 42, 44, 49, 54]	[15, 16, 58–60, 61, 63, 64]
6 Type of application	6.1 Time based	[12, 28, 31, 37, 42–57]	[15, 16, 58–60, 62–67]
	6.2 Event based	[28, 31, 42, 44, 48, 50, 52, 55, 56]	[64]
7 Optimization problem formulation	7.1 Integer	[12, 50, 52, 54–57]	[15, 16, 62, 63]
	7.2 Linear	[12, 49, 50, 57]	[15, 16, 59, 63]
	7.3 Nonlinear		[62]
	7.4 Multi-objective	[28, 42, 51, 57]	
8 Approach of the algorithm proposed/used to solve the problem	8.1 Graph theory	[12, 28, 31, 47, 57]	
	8.2 Evolutionary	[42, 46, 47, 51]	
	8.3 Game /auction	[48, 52, 55]	[61, 67]
	8.4 Greedy	[12, 28, 31, 37, 43, 45, 49, 50, 53, 54, 56, 57]	[60, 63]
	8.5 Machine learning	[46]	
	8.6 Voronoi Diagram	[44]	
	8.7 Lin. programming		[15, 16, 59, 62]
	8.8 Probabilistic		[65, 66]
9 Considers the edge tier		[42]	[62, 64, 65, 67]
10 Devices as active resource providers		[28, 57]	[63–67]

cycles. In such partially dynamic approaches, the state of the system (acquired at the beginning of each cycle and used for making the scheduling/allocation decision) is assumed to remain the same during the entire duration of that cycle. This is not adequate when considering applications that handle continuous data streams [50]. To support this kind of application, it is necessary to model the state of each resource as a function that varies continuously in time, thus fully handling the dynamic and continuous characteristic of the CoS environment, as in [50].

Therefore, there are several open issues in the current literature not fully addressed by any of the proposals. There is a lack of a decentralized full device virtualization model, with devices playing an active role as resource providers and considering the edge tier in its architecture. In addition, this virtualization model should perform resource allocation and task scheduling, formulated as optimization problems, through effective, fast, and lightweight algorithms. Finally, this virtualization model should handle priorities and precedencies among requests and tasks, share requests and tasks in common among multiple applications and VN programs, and support time-based and event-based applications simultaneously. In the following section, we present a brief overview of our own approach of a full device virtualization model, to perform both resource allocation and task scheduling in CoS. This approach is based on the Olympus virtualization model, described in our previous work [10].

4 Resource Allocation and Task Scheduling in Olympus

In this section, we present a brief overview of our own approach of a device virtualization model, to be used as part of an integrated solution for both resource allocation and task scheduling in CoS. First, we briefly describe the Olympus virtualization model [10], presenting a general view of our virtualization model. Next, we discuss how Olympus addresses some of the open issues raised in this chapter.

Olympus is a decentralized WSAN virtualization model for CoS. It seeks to make the best use of the cloud and the physical WSAN environments by finding a balance between two possible approaches for running services: centrally, inside the cloud, and locally, within the physical sensors. Olympus leverages the use of information fusion to ensure that the system will provide data at a given abstraction level more suitable to user applications. Olympus is a decentralized virtualization model since the physical nodes can locally perform the necessary procedures for creating and running the virtual sensor. Therefore, in Olympus, application decision processes are performed partly within physical sensors and partly within the cloud. Olympus abstracts the physical world, by abstracting issues regarding the spatial/geographical distribution of each sensor at the physical layer. Over the physical layer, there is the information fusion layer, based on the information fusion levels according to the classifications of the data-feature-decision (DFD) information fusion model [68].

In Olympus, we make use of the data-feature-decision (DFD) model that provides a classification for information fusion techniques according to the abstraction of the input and output data. In Data In–Data Out (DAI-DAO), information fusion deals

with measurement level data as input and output. In Data In-Feature Out (DAI-FEO), information fusion uses data at the measurement level as input to extract attributes or characteristics that describe more summarized information for a given monitored area. In the Feature In-Feature Out (FEI-FEO) category, information fusion works on a set of features to improve or refine an existing characteristic or attribute, or to extract new ones. In the Feature In-Decision Out (FEI-DEO) category, information fusion uses a number of features extracted for generating a symbolic representation (or a decision). In the Decision In-Decision Out (DEI-DEO) category, decisions can be merged to obtain new decisions. Finally, in the Data In-Decision Out (DAI-DEO), either a decision is made directly over raw data, as an atomic procedure, or the information fusion process under this category can be broken into several atomic parts pertaining to other categories.

In Olympus, an application is considered as a set of services that must be performed to accomplish the application goals. Each application has a finite life span and it is interested in a particular geographical area. An application defines a set of QoS requirements, described in terms of maximum end-to-end delay, maximum percentage of packet loss, and energy consumption. Moreover, applications require a set of services provided by the physical WSAN nodes that are described in terms of the following provided services: (i) Data collection, (ii) Processing, (iii) Decision, (iv) Routing, and (v) Actuation. Such capabilities must be published within the cloud in a central repository through a publish/subscribe mechanism. However, physical sensors connected to the CoS must have the minimal capability of locally (in its physical location) providing continuous raw data at a periodic rate. This premise is less restrictive than the works proposing centralized CoS infrastructures support, in which the physical nodes must provide such raw data directly to the sink node.

For connecting applications to physical WSAN nodes, one or more virtual WSAN must be created. A virtual WSAN is created by providing logical connectivity among the physical nodes. Such physical nodes are grouped into different virtual WSANs based on the phenomenon being monitored or the service being provided. A virtual WSAN node in Olympus is an abstraction of a set of physical nodes, from which the virtual node obtains data. Such a virtual node is considered a computational entity capable of performing a set of information fusion techniques at a given level of DFD model, as shown in Fig. 4. Virtual nodes may also form logical neighborhoods. In contrast with physical neighborhoods, usually defined in terms of radio ranges, the nodes included in a logical neighborhood are specified by the application based on specific requirements.

In Olympus, there is a computational entity, which we term the Virtualization Manager. This entity runs within the cloud and has several responsibilities regarding the model execution management. Our model is said to be partly decentralized because the services allocated by this entity will run within the physical nodes.

Each VN in Olympus represents the implementation of an information fusion procedure, which can be reused by several applications. In Olympus, the VNs depend not only on the information fusion level of the input/output data, but also on the source of such data within the physical network. Olympus supports traditional (i) one-to-one, (ii) one-to-many, and (iii) many-to-one virtualization schemes. Moreover, it supports

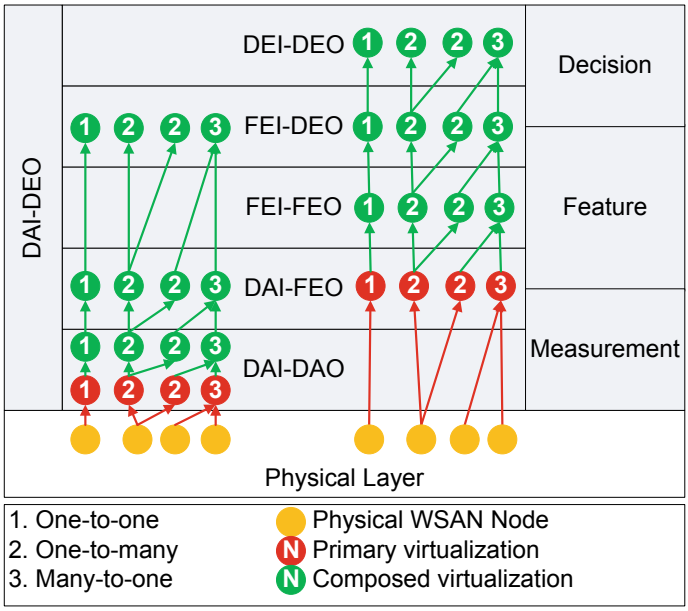


Fig. 4 Linking VNs and data abstraction levels of information fusion in Olympus

primary virtualization (a virtualization performed when a first VN is instantiated directly from a PSAN), as well as composed virtualizations (in which the virtual sensors are created from another VN).

Olympus operation model comprises both sequential phases: (i) VN creation (instantiation) and (ii) VN operation. In the first phase, Olympus considers a publish/subscribe mechanism to choose the proper PSANs that meet the requirements of the applications requesting the creation of the VN. The Virtualization Manager is responsible to read the application requirements (through subscriptions) and the PSAN capabilities published in cloud, for deciding if a new instance of a VN should be created, or an existing instance should be reused. The PSANs, on receiving any request (sent by the Virtualization Manager) to start the instantiation of a VN, are elected as leaders. It is the responsibility of such leader nodes (elected by the Virtualization Manager) to search for and establish routes for communicating with other physical sensors (possibly in other separate physical WSAN) required by a given virtual sensor to be created. Leader nodes will be the reference nodes for the Virtualization Manager to communicate with when retrieving data or performing procedures for VN creation and operation management. Leader nodes are the ones that perform the information fusion techniques at the highest level required by the instantiated VN. Therefore, VN creation is performed partly within the cloud and partly within the PSANs.

When the second phase starts, all PSANs are ready for performing any application request allocated to the respective VNs. In this phase, the Virtualization Manager

allocates the execution of the application requests within the PSANs (service allocation). During operation, if an undesired state is detected, the Virtualization Manager will issue warnings to the application users through push communication. However, every physical sensor must be ready to perform pull communication during the virtual sensor operation management. This is because the application users may want to stay up to date with the current execution status of the allocated services. Olympus supports both time-based and event-based applications simultaneously. If any PSAN pertains to a separate WSANI, then the routing among the PSANs must go through the sink nodes of both WSANIs to enable passage through the cloud. One first approach to resolve this issue in Olympus is to treat the issue as a routing problem. The second possibility is equivalent to the approach of a centralized CoS virtualization model. That is to say, that the PSANs send data to the cloud through the sink nodes, and the VN creation takes place only within the cloud. Finally, the VN operation is more prone to run within PSANs than VN instantiation. VN may even operate without any communication with the cloud. However, there are still procedures of VN operation that may be performed partly within the cloud, such as the case of many-to-one virtualization comprising nodes physically separated in different WSANs. In face of the presented discussions, Olympus is considered a hybrid (partly decentralized) WSAN virtualization model.

By analyzing the first version of Olympus, we identified some open issues, and several aspects that were not described in depth in our previous paper [10]. Olympus is not a fully decentralized virtualization model; however, it is a full device virtualization model, with devices playing an active role as resource providers. Moreover, Olympus does not consider the edge tier in its architecture. Although sink nodes play an important role in Olympus for performing communication among PSANs and VNs, there is no clear scheme proposed for using the computation capabilities at the edge of the network. In addition, Olympus has no description of mechanisms for performing resource allocation and task scheduling, formulated as optimization problems, through effective, fast, and lightweight algorithms. Moreover, Olympus does not handle priorities and precedencies among requests and tasks and does not share requests and tasks in common among multiple applications and VN programs. Finally, there is support to time-based and event-based applications simultaneously in Olympus, but these kinds of applications are not modeled in depth in Olympus. In future works, we aim at providing solutions to tackle each of the open issues mentioned in this section.

5 Final Remarks

In this chapter, the challenges regarding the development of solutions for performing resource allocation and task scheduling in the CoS environment were discussed. We depicted the background concepts of resource allocation and task scheduling in CoS. Moreover, we described several solutions found in the literature that we consider being of utmost importance for understanding the context of task scheduling and

resource allocation in CoS. In addition, we pointed out several open issues in the state of the art and presented our own solution for CoS virtualization, assessing it in face of each open issue. We consider that the CoS paradigm is of great relevance and deserves discussions, because it is going to be the future trend for large-scale WSAN deployments integrated to the Internet. Finally, in the context of mission-oriented WSANs, the CoS architecture shows great potential of contribution. The CoS is a generic and shared infrastructure, in which several virtual WSANs can exist. In this scenario, each virtual WSAN can be oriented to a mission and, thus, can be created on demand, providing the scalability and flexibility required by the typical mission-oriented paradigm.

Acknowledgements This work is partly supported by the following Brazilian funding agencies: National Council for Scientific and Technological Development (CNPq), Financier of Studies and Projects (FINEP), and the Foundation for Research of the State of Rio de Janeiro (FAPERJ).

References

1. Aleksic, S.: Green ICT for sustainability: a holistic approach. In: 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), May, pp. 426–431 (2014). https://doi.org/10.1109/mipro.2014.6859604
2. Atzori, L., Iera, A., Morabito, G.: The Internet of Things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010). https://doi.org/10.1016/j.comnet.2010.05.010
3. Botta, A., et al.: On the integration of cloud computing and Internet of Things. In: International Conference on Future IoT and Cloud (FiCloud), pp. 23–30 (2014)
4. Rawat, P., Singh, K.D., Chaouchi, H., Bonnin, J.M.: Wireless sensor networks: a survey on recent developments and potential synergies. J. Supercomput. **68**(1), 1–48 (2013). https://doi.org/10.1007/s11227-013-1021-9
5. Alamri, A., Ansari, W.S., Hassan, M.M., Hossain, M.S., Alelaiwi, A., Hossain, M.A.: A survey on sensor-cloud: architecture, applications, and approaches. Int. J. Distrib. Sens. Netw. **2013**, 1–18 (2013). http://doi.org/10.1155/2013/917923
6. Zeng, D., Gu, L., Guo, S., Cheng, Z., Yu, S.: Joint optimization of task scheduling and image placement in Fog computing supported software-defined embedded system. IEEE Trans. Comput. **PP**(99), 1–1 (2016). http://doi.org/10.1109/TC.2016.2536019
7. Yi, S., Li, C., Li, Q.: A survey of Fog computing. In: Proceedings of the 2015 Workshop on Mobile Big Data—Mobidata '15, pp. 37–42. ACM Press, New York, New York, USA (2015). http://doi.org/10.1145/2757384.2757397
8. Li, W., Santos, I., Delicato, F.C., Pires, P.F., Pirmez, L., Wei, W., Khan, S.: System modelling and performance evaluation of a three-tier Cloud of Things. Future Gener. Comput. Syst. (2016). https://doi.org/10.1016/j.future.2016.06.019
9. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the Internet of Things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, pp. 13–16 (2012). http://doi.org/10.1145/2342509.2342513
10. Santos, I.L., Pirmez, L., Delicato, F.C., Khan, S.U., Zomaya, A.Y.: Olympus: the cloud of sensors. IEEE Cloud Comput. **2**(2), 48–56 (2015). https://doi.org/10.1109/MCC.2015.43
11. Islam, M.M., Hassan, M.M., Lee, G.-W., Huh, E.-N.: A survey on virtualization of wireless sensor networks. Sensors **12**(12), 2175–2207 (2012). https://doi.org/10.3390/s120202175
12. Rowaihy, H., Johnson, M.P., Liu, O., Bar-Noy, A., Brown, T., Porta, T.La.: Sensor-mission assignment in wireless sensor networks. ACM Trans. Sens. Netw. **6**(4), 1–33 (2010). https://doi.org/10.1145/1777406.1777415

13. Preece, A., Braines, D., Pizzocaro, D., Parizas, C.: Human-machine conversations to support mission-oriented information provision. In: Proceedings of the 2nd ACM Annual International Workshop on Mission-Oriented Wireless Sensor Networking—MiSeNet '13, p. 43. ACM Press, New York, New York, USA (2013). http://doi.org/10.1145/2509338.2509342
14. Santos, L., Pirmez, L., Carmo, L.R., Pires, P.F., Delicato, F.C., Khan, S.U., Zomaya, A.Y.: A decentralized damage detection system for wireless sensor and actuator networks. IEEE Trans. Comput. **65**, 1363–1376 (2016). https://doi.org/10.1109/tc.2015.2479608
15. Delgado, C., Gallego, J.R., Canales, M., Ortin, J., Bousnina, S., Cesana, M.: An optimization framework for resource allocation in virtual sensor networks. In: 2015 IEEE Global Communications Conference (GLOBECOM), pp. 1–7. IEEE (2015). http://doi.org/10.1109/GLOCOM.2015.7417187
16. Delgado, C., Gállego, J.R., Canales, M., Ortín, J., Bousnina, S., Cesana, M.: On optimal resource allocation in virtual sensor networks. Ad Hoc Netw. **50**, 23–40 (2016). https://doi.org/10.1016/j.adhoc.2016.04.004
17. De Farias, C.M., Li, W., Delicato, F.C., Pirmez, L., Zomaya, A.Y., Pires, P.F., De Souza, J.N.: A systematic review of shared sensor networks. ACM Comput. Surv. **48**(4), 1–50 (2016). https://doi.org/10.1145/2851510
18. Sinnen, O.: Task scheduling for parallel systems. In: Wiley Series on Parallel and Distributed Computing (2007). https://doi.org/10.1002/9780470121177.scard
19. Iman, M., Delicato, F.C., de Farias, C.M., Pirmez, L., dos Santos, I.L., Pires, P.F.: THESEUS: a routing system for shared sensor networks. In: 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, pp. 108–115. IEEE (2015). http://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.18
20. Eltarras, R., Eltoweissy, M.: Adaptive multi-criteria routing for shared sensor-actuator networks. In: GLOBECOM—IEEE Global Telecommunications Conference (2010). https://doi.org/10.1109/GLOCOM.2010.5683555
21. Barbaran, J., Diaz, M., Rubio, B.: A virtual channel-based framework for the integration of wireless sensor networks in the cloud. In: 2014 International Conference on Future Internet of Things and Cloud, pp. 334–339. IEEE (2014). http://doi.org/10.1109/FiCloud.2014.59
22. Saha, S.: Secure sensor data management model in a sensor-cloud integration environment. In: 2015 Applications and Innovations in Mobile Computing (AIMoC), pp. 158–163. IEEE (2015). http://doi.org/10.1109/AIMOC.2015.7083846
23. Das, C., Tripathy, S.P.: A review on virtualization in wireless sensor network. **1**(1), 1–8 (2010)
24. Khan, I., Belqasmi, F., Glitho, R., Crespi, N., Morrow, M., Polakos, P.: Wireless sensor network virtualization: a survey. IEEE Commun. Surv. Tutor. **18**(1), 553–576 (2016). https://doi.org/10.1109/COMST.2015.2412971
25. Bouzeghoub, M.: A framework for analysis of data freshness. In: Proceedings of the 2004 International Workshop on Information Quality in Informational Systems—IQIS '04, vol. 59 (2004). http://doi.org/10.1145/1012453.1012464
26. Gonçalves, B., Filho, J.G.P., Guizzardi, G.: A service architecture for sensor data provisioning for context-aware mobile applications. In: Proceedings of the 2008 ACM Symposium on Applied Computing—SAC '08, vol. 1946 (2008). http://doi.org/10.1145/1363686.1364155
27. Yang, R., Wo, T., Hu, C., Xu, J., Zhang, M.: Dˆ2PS: a dependable data provisioning service in multi-tenant cloud environment. In: 2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE), pp. 252–259 (2016). http://doi.org/10.1109/HASE.2016.26
28. Li, W., Delicato, F.C., Pires, P.F., Lee, Y.C., Zomaya, A.Y., Miceli, C., Pirmez, L.: Efficient allocation of resources in multiple heterogeneous wireless sensor networks. J. Parallel Distrib. Comput. **74**(1), 1775–1788 (2014). https://doi.org/10.1016/j.jpdc.2013.09.012
29. Bauer, M., Bui, N., Jardak, C., Nettsträter, A.: The IoT ARM reference manual. In: Bassi, A., Bauer, M., Fiedler, M., Kramp, T., van Kranenburg, R., Lange, S., Meissner, S. (eds.) Enabling Things to Talk, pp. 213–236. Springer, Berlin, Heidelberg (2013). http://doi.org/10.1007/978-3-642-40403-0_9

30. Gmach, D., Rolia, J., Cherkasova, L., Kemper, A.: Resource pool management: reactive versus proactive or let's be friends. Comput. Netw. **53**(17), 2905–2922 (2009). https://doi.org/10.1016/j.comnet.2009.08.011
31. de Farias, C.M., Pirmez, L., Delicato, F.C., Li, W., Zomaya, A.Y., de Souza, J.N.: A scheduling algorithm for shared sensor and actuator networks. In: The International Conference on Information Networking 2013 (ICOIN), pp. 648–653 (2013). http://doi.org/10.1109/ICOIN.2013.6496703
32. Gonçalves, G.E., Endo, P.T., Cordeiro, T.D., Palhares, André Vitor de Almeida Sadok, D., Kelner, J., Melander, B., Mångs, J.-E.: Resource allocation in clouds: concepts, tools and research challenges. Minicursos - XXIX Simpósio Brasileiro de Redes de Computadores E Sistemas Distribuídos, pp. 197–240 (2011). http://www.cricte2004.eletrica.ufpr.br/anais/sbrc/2011/files/mc/mc5.pdf
33. Liu, R., Wassell, I.J.: Opportunities and challenges of wireless sensor networks using cloud services. In: Proceedings of the Workshop on Internet of Things and Service Platforms—IoTSP '11, pp. 1–7. ACM Press, New York, New York, USA (2011). https://doi.org/10.1145/2079353.2079357
34. Aleksic, S.: Green ICT for sustainability: a holistic approach. In: 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), May, pp. 426–431 (2014). https://doi.org/10.1109/mipro.2014.6859604
35. Open Geospatial Consortium [Online]. http://www.opengeospatial.org/
36. Zhang, J., Huang, H., Wang, X.: Resource provision algorithms in cloud computing: a survey. J. Netw. Comput. Appl. **64**, 23–42 (2016). https://doi.org/10.1016/j.jnca.2015.12.018
37. Kaur Kapoor, N., Majumdar, S., Nandy, B.: Techniques for allocation of sensors in shared wireless sensor networks. J. Netw. **10**(1), 15–28 (2015). https://doi.org/10.4304/jnw.10.1.15-28
38. Raghavendra, P.: Approximating NP-Hard Problems Efficient Algorithms and Their Limits (2009)
39. Bolaños, R.I., Echeverry, M.G., Escobar, J.W.: A multiobjective non-dominated sorting genetic algorithm (NSGA-II) for the multiple traveling salesman problem. Decis. Sci. Lett. **4**(4), 559–568 (2015). https://doi.org/10.5267/j.dsl.2015.5.003
40. Vanderbei, R.J.: Linear programming: foundations and extensions. J. Oper. Res. Soc. **49**(1), 94–94 (1998). http://doi.org/10.1057/palgrave.jors.2600987
41. Garey, M.R., Johnson, D.S.: Computers and intractability (1979)
42. Phan, D., Suzuki, J., Omura, S., Oba, K.: Toward sensor-cloud integration as a service: optimizing three-tier communication in cloud-integrated sensor networks. In: Proceedings of the 8th International Conference on Body Area Networks, vol. 1. ACM (2013). http://doi.org/10.4108/icst.bodynets.2013.253639
43. Zhu, C., Li, X., Leung, V.C.M., Hu, X., Yang, L.T.: Job scheduling for cloud computing integrated with wireless sensor network. In: 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, pp. 62–69 (2014). http://doi.org/10.1109/CloudCom.2014.106
44. Dalvi, R.: Energy efficient scheduling and allocation of tasks in sensor cloud (2014)
45. Yao, R., Wang, W., Shin, S., Son, S.H., Jeon, S.I.: Competition-based device-to-device transmission scheduling to support wireless cloud multimedia communications. Int. J. Distrib. Sens. Netw. **2014** (2014). http://doi.org/10.1155/2014/869514
46. Li, J., Pan, Q., Mao, K.: Solving complex task scheduling by a hybrid genetic algorithm, pp. 3440–3443 (2014)
47. Kim, M., Ko, I.Y.: An efficient resource allocation approach based on a genetic algorithm for composite services in IoT environments. In: Proceedings—2015 IEEE International Conference on Web Services, ICWS 2015, pp. 543–550 (2015). http://doi.org/10.1109/ICWS.2015.78
48. Kim, S.: Asymptotic shapley value based resource allocation scheme for IoT services. Comput. Netw. **100**, 55–63 (2016). https://doi.org/10.1016/j.comnet.2016.02.021

49. Li, L., Li, S., Zhao, S.: QoS-aware scheduling of services-oriented Internet of Things. IEEE Trans. Ind. Inf. **10**(2), 1497–1505 (2014). https://doi.org/10.1109/TII.2014.2306782
50. Billet, B., Issarny, V.: From task graphs to concrete actions: a new task mapping algorithm for the future Internet of Things. In: 2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems, pp. 470–478. IEEE (2014). http://doi.org/10.1109/MASS.2014.20
51. Dai, L., Xu, H.K., Chen, T., Qian, C., Xie, L.J., Chen, T.: A multi-objective optimization algorithm of task scheduling in WSN. **9**(2), 160–171 (2014)
52. Wu, C., Xu, Y., Chen, Y., Lu, C.: Submodular game for distributed application allocation in shared sensor networks. In: Proceedings—IEEE INFOCOM, pp. 127–135 (2012). http://doi.org/10.1109/INFCOM.2012.6195490
53. Wang, F., Han, G., Jiang, J., Qiu, H.: A distributed task allocation strategy for collaborative applications in cluster-based wireless sensor networks. Int. J. Distrib. Sens. Netw. **2014**(i), 1–16 (2014). http://doi.org/10.1155/2014/964595
54. Hu, W., O'Rourke, D., Kusy, B., Wark, T.: A virtual sensor scheduling framework for heterogeneous wireless sensor networks. In: 38th Annual IEEE Conference on Local Computer Networks, pp. 655–658. IEEE (2013). http://doi.org/10.1109/LCN.2013.6761303
55. Edalat, N., Xiao, W., Roy, N., Das, S.K., Motani, M.: Combinatorial auction-based task allocation in multi-application wireless sensor networks. In: 2011 IFIP 9th International Conference on Embedded and Ubiquitous Computing, pp. 174–181. IEEE (2011). http://doi.org/10.1109/EUC.2011.22
56. Bhattacharya, S., Saifullah, A., Lu, C., Roman, G.-C.: Multi-application deployment in shared sensor networks based on quality of monitoring. In: 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium, vol. 4, pp. 259–268. IEEE (2010). http://doi.org/10.1109/RTAS.2010.20
57. Li, W., Delicato, F.C., Zomaya, A.Y.: Adaptive energy-efficient scheduling for hierarchical wireless sensor networks. ACM Trans. Sens. Netw. **9**(3), 1–34 (2013). https://doi.org/10.1145/2480730.2480736
58. Misra, S., Chatterjee, S., Obaidat, M.S.: On theoretical modeling of sensor cloud: a paradigm shift from wireless sensor network. IEEE Syst. J. 1–10 (2014). http://doi.org/10.1109/JSYST.2014.2362617S
59. Dinh, T., Kim, Y.: An efficient interactive model for on-demand sensing-as-a-services of sensor-cloud. Sensors **16**(7), 992 (2016). https://doi.org/10.3390/s16070992
60. Narman, H.S., Hossain, M.S., Atiquzzaman, M., Shen, H.: Scheduling internet of things applications in cloud computing. Ann. Telecommun. (2016). https://doi.org/10.1007/s12243-016-0527-6
61. Yu, R., Zhang, Y., Gjessing, S., Xia, W., Yang, K.: Toward cloud-based vehicular networks with efficient resource management. IEEE Netw. **27**(5), 48–55 (2013). https://doi.org/10.1109/MNET.2013.6616115
62. Zeng, D., Gu, L., Guo, S., Cheng, Z., Yu, S.: Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. IEEE Trans. Comput. **PP**(99), 1–1 (2016). http://doi.org/10.1109/TC.2016.2536019
63. Angelakis, V., Avgouleas, I., Pappas, N., Yuan, D.: Flexible allocation of heterogeneous resources to services on an IoT device. In: Proceedings—IEEE INFOCOM, 2015–Augus(5), 99–100 (2015). http://doi.org/10.1109/INFCOMW.2015.7179362
64. Vögler, M., Schleicher, J.M., Inzinger, C., Nastic, S., Sehic, S., Dustdar, S.: LEONORE—large-scale provisioning of resource-constrained IoT deployments. In: 9th International Symposium on Service-Oriented System Engineering (2015). http://doi.org/10.1109/SOSE.2015.23
65. Aazam, M., St-Hilaire, M., Lung, C.H., Lambadaris, I.: PRE-Fog: IoT trace based probabilistic resource estimation at Fog. In: 2016 13th IEEE Annual Consumer Communications and Networking Conference, CCNC 2016, pp. 12–17 (2016). http://doi.org/10.1109/CCNC.2016.7444724
66. Aazam, M., Huh, E.-N.: Resource management in media cloud of things. In: 2014 43rd International Conference on Parallel Processing Workshops, vol. 2015–May, pp. 361–367. IEEE (2014). http://doi.org/10.1109/ICPPW.2014.54

67. Abedin, S.F., Alam, M.G.R., Tran, N.H., Hong, C.S.: A Fog based system model for cooperative IoT node pairing using matching theory. In: 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 309–314. IEEE (2015). http://doi.org/10.1109/APNOMS.2015.7275445
68. Nakamura, E.F., Loureiro, A.A.F., Frery, A.C.: Information fusion for wireless sensor networks. ACM Comput. Surv. **39**(3), 9–es (2007). http://doi.org/10.1145/1267070.1267073

Part IV
Detection, Localization, and Tracking

Target Detection, Localization, and Tracking in Wireless Sensor Networks

Jing Liang, Xiaofeng Yu, Xiaoxu Liu, Chengchen Mao and Jie Ren

Abstract This chapter will investigate target detection approaches, sensor node localization algorithms, and target tracking schemes in wireless sensor network (WSN). WSN is a self-organized distributed network composed of a large quantity of small, inexpensive sensor nodes. They are employed to capture information of the target, which is similar to humans' sense of hearing, sight, smell, and touch. The integration of a plurality of homogeneous or heterogeneous sensors will result in more accurate target detection, node localization, and target tracking than that of a single sensor, and thus WSN plays an important role in military, environment, medical, and industrial fields.

1 Target Detection in Wireless Sensor Networks

In a target detection application, sensor nodes transmit waveforms of known shapes and receive the echoes from targets and various obstacles. Some researchers name this type of nodes "radar sensor," and their corresponding network "radar sensor network (RSN)" [1–8]. RSNs can be utilized to combat the performance of signal degradation in wireless channels. They are arranged to survey a large area and observe targets from a number of different angles.

A RSN not only provides spatial resilience for target detection and tracking compared to traditional radars, but also alleviates inherent radar defects such as the blind speed problem. This interdisciplinary area offers a new paradigm for parallel and distributed sensor research.

J. Liang (✉) · X. Yu · X. Liu · C. Mao · J. Ren
University of Electronic Science and Technology of China, Chengdu, China
e-mail: liangjing@uestc.edu.cn

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_9

1.1 Coherent and Noncoherent WSN Detection Systems

In this section, we mainly focus on both coherent and noncoherent RSN detection systems applying selection combination algorithm (SCA) performed by a leading sensor to take the advantage of spatial diversity. We will also analyze the impact of Doppler shift on both coherent and noncoherent RSN detection systems at the presence of clutter.

1.1.1 Distributed Model and Problem Formulation

A RSN incorporates N radar sensors working in a self-organizing fashion. Each sensor can detect targets and provide the detected signals to their clusterhead (CH) sensor, which combines these waveforms and makes the final decision of target detection. We assume there is no information loss when transmitting signals to the CH. The propagation and target model of a RSN are illustrated in Fig. 1. Complex target signals are constructed from distinct scatterers. The radar cross section (RCS) fluctuates when the target changes relatively to the radar antenna [9]. In this case, the RCS is usually presented by Rayleigh probability density function [10, 11]. As the amplitude of each pulse is statistically independent, the "Swerling II" model can be applied for a pulse-to-pulse fluctuating target.

For clarity and simplicity, we apply a constant frequency (CF) impulse f_c with the same pulse duration T_p to each sensor. Every transmitted impulse consists of a sinusoidal waveform that is typically expressed as

$$\tilde{S}_i(t) = A_{ti} \cdot \sqrt{\frac{2}{T_p}} \cos\left[2\pi(f_c + \Delta_i)(t + t_i)\right]. \tag{1}$$

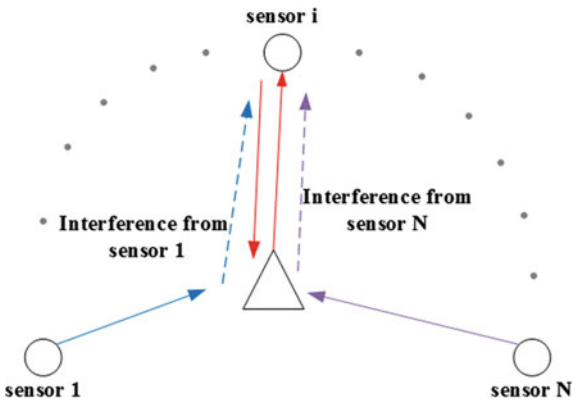


Fig. 1 Propagation and target model for RSNs

Assume t_i seconds after transmitting the pulse, the received combined back-scattered signal can be modeled as

$$\tilde{R}_i(t) = \tilde{S}_{ri}(t) + \tilde{I}_i(t) + \tilde{C}_i(t) + n_{ri}(t), \tag{2}$$

where $\tilde{S_{ri}}(t)$ is the expected back-scattered radiation

$$\tilde{S}_{ri}(t) = A_i \cdot \sqrt{\frac{2}{T_p}} \cos\left[2\pi(f_c + \Delta_i + f_{di})t)\right]. \tag{3}$$

A_i represents the amplitude of the returned radar waveform, and f_{di} denotes the Doppler shift in the returned signal compared to the transmitted waveform.

As the Swerling II model is applied, $|A_i|$ is a random variable that follows a Rayleigh distribution, which can be denoted as $A_i = A_i^I + jA_i^Q$, and both I and Q subchannels of A_i are Gaussian distributed with zero mean and variance $\gamma^2/2$.

Assume the target is moving at a speed v, as each sensor provides a unique carrier frequency and location to the same target, f_{di} can be given as

$$f_{di} = 2 \cdot \frac{v(f_c + \Delta_i)}{c} \cdot \cos\phi = f_{di_{max}} \cdot \cos\phi, \tag{4}$$

where c is the speed of light, and ϕ is the elevation angle between each sensor and the target. Normally, a RSN can be deployed on high mountains or low ground, therefore a target can be above or below a RSN. We may consider a RSN uniformly distributed around the target, and thus ϕ is a random variable that follows a uniform distribution within $[0, 2\pi]$, owning to the uncertainty of this angle.

When all of radar sensors are working, sensor i not only receives its own back-scattered waveform, but also scattered signals generated by other sensor. These interfering waveforms received by sensor i can be modeled as

$$\tilde{I}_i(t) = \sum_{k=1,k\neq i}^{N} B_k \cdot \sqrt{\frac{2}{T_p}} \cos\left[(f_c + \Delta_k + f_{dk})t\right]. \tag{5}$$

where $B_k = B_k^I + jB_k^Q$ is the interference from radar k. B_k^I and B_k^Q can be effectively approximated by two zero-mean Gaussian distributions, each with variance $\rho^2/2$. Therefore, similar to $|A_i|$, $|B_k|$ also follows Rayleigh distribution, and f_{dk} is the Doppler shift based on geometric configuration of sensor i, sensor k, and the target.

As far as the clutter is concerned, $\tilde{C}_i(t)$ can be given as

$$\tilde{C}_i(t) = M_i \cdot \sqrt{\frac{2}{T_p}} \cos\left[2\pi(f_c + \Delta_i)t\right]. \tag{6}$$

Similarly, $C_i = C_i^I + jC_i^Q$, where C_i^I and C_i^Q subchannels are Gaussian distributed with zero mean and variance $\eta^2/2$, respectively. Apart from the clutter, the radar i also receives additive white Gaussian noise (AWGN) $n_{ri}(t) = n_{ri}^I(t) + jn_{ri}^Q(t)$, where I and Q subchannels follow zero-mean Gaussian distributions with variance $\sigma^2/2$. After introducing our propagation and target model, further analysis on coherent and noncoherent RSNs is carried out.

1.1.2 Coherent Detection

In coherent RSNs, radar members are smart enough to obtain the knowledge of the exact Doppler shift introduced by moving targets. The output of the ith branch $Y_i(t)$ is

$$Y_i = \int_0^{T_p} \tilde{R}_i(t) \cdot \sqrt{\frac{2}{T_p}} \cos\left[2\pi(f_c + \Delta_i + f_{di})t\right] dt = S_i + I_i + C_i + n_i. \tag{7}$$

where S_i, I_i, C_i, and n_i denote the output of useful signal, interference, clutter, and noise, respectively. Similarly, the output envelope of sensor i can be represented as

$$|Y_i| \approx \left| A_i + \sum_{k=1,k\neq i}^{N} \frac{B_k \sin\left[2\pi(f_{dk} - f_{di})T_p\right]}{2\pi\left[(k-i) + (f_{dk} - f_{di})T_p\right]} + M_i + n_i \right|, \tag{8}$$

where $S_i = A_i$ and $C_i \approx M_i$. n_i follows Gaussian distribution with variance $\sigma^2/2$.

To simplify the expression, we define

$$e = E\left\{ \frac{sin\left[2\pi(f_{dk} - f_{di})T_p\right]}{2\pi\left[(k-i) + (f_{dk} - f_{di})T_p\right]} \right\}. \tag{9}$$

Here $E\{\}$ denotes the expectation. Therefore, (8) becomes

$$|Y_i| \approx \left| A_i + \sum_{k=1,k\neq i}^{N} eB_k + M_i + n_i \right| = \left| A_i + \sum_{k=1,k\neq i}^{N} eB_k^I + j \sum_{k=1,k\neq i}^{N} eB_k^Q + M_i + n_i \right|, \tag{10}$$

$\sum_{k=1,k\neq i}^{N} eB_k^I$ and $\sum_{k=1,k\neq i}^{N} eB_k^Q$ follow Gaussian distributions with the same variance $\frac{\beta^2}{2} = (N-1)\frac{e^2\rho^2}{2}$. Therefore, $\left|\sum_{k=1,k\neq i}^{N} eB_k\right|$ follows a Rayleigh distribution. Since $|A_i|$, M_i and $|n_i|$ are also Rayleigh random variables, $|Y_i|$ follows a Rayleigh distribution with the parameter

$$\alpha = \sqrt{\gamma^2 + \beta^2 + \eta^2 + \sigma^2}. \tag{11}$$

To this end when there is a moving target, the p.d.f for $|Y_i|$ is

$$f_s(y_i) = \frac{y_i}{\alpha^2} exp\left(-\frac{y_i^2}{2\alpha^2}\right). \tag{12}$$

The mean value of y_i is $\alpha\sqrt{\frac{\pi}{2}}$, and the variance is $\left(2-\frac{\pi}{2}\right)\alpha^2$. The variance of useful radar signal, clutter, and noise are $\left(2-\frac{\pi}{2}\right)\gamma^2$, $\left(2-\frac{\pi}{2}\right)\eta^2$, $\left(2-\frac{\pi}{2}\right)\sigma^2$, respectively. Therefore, SNR is $\frac{\gamma^2}{\sigma^2}$ and signal-to-clutter ratio is $\frac{\gamma^2}{\eta^2}$.

Before making a final decision, the RSN CH applies a selection combination algorithm to take the advantage of spatial diversity. Of course, other combining schemes, such as the equal gain combining algorithm (EGCA), can also be applied. The combining schemes are out of the scope of this subsection. Readers may refer to [32] for more details. The combiner selects the sensor with the maximum envelope. This is equivalent to choosing the sensor with highest $\frac{\gamma^2}{\sigma^2}$ and $\frac{\gamma^2}{\eta^2}$. Therefore, the output of the combiner has an SNR equal to the maximum SNR of all the sensors. With SCA, since only one branch output is used, co-phasing of multiple branches is not required and therefore, it can be used in both coherent and noncoherent RSNs.

On account of independence of each $|Y_i|$, the p.d.f. of the output from diversity combiner is

$$f_s(y) = \prod_{i=2}^{N} \frac{y_i}{\alpha^2} exp\left(-\frac{y^2}{2\alpha^2}\right). \tag{13}$$

In case of no target, i.e., there exist only clutter and noise, the p.d.f. of output from diversity combiner becomes

$$f_{cn}(y) = \prod_{i=1}^{N} \frac{y_i}{\zeta^2} exp\left(-\frac{y_i^2}{2\zeta^2}\right). \tag{14}$$

where $\zeta = \sqrt{\eta^2 + \sigma^2}$.

In light of the p.d.f. for the above two cases, we apply Bayesian rule to decide the existence of targets based on y

$$\frac{f_s(y)}{f_{cn}(y)} \underset{\text{no target}}{\overset{\text{target exisits}}{\gtrless}} \frac{P_{cn}}{P_s} \tag{15}$$

where P_{cn} denotes the probability of no target but noise, and P_s represents the probability of target occurrence.

1.1.3 Noncoherent Detection

As far as a noncoherent RSN is concerned, its difference from the above system is that radar sensors have no knowledge of exact Doppler shift in back-scattered signals, so each matched filter applies the same frequency as that of transmitted waveforms,

and finally leads to more ambiguity in target detection. In spite of its complexity, this system is more practical.

Consider the sensor i, the output of inphase branch and quadrature branch are

$$Y_i^I = \int_0^{T_p} \tilde{R}_i(t) \cdot \sqrt{\frac{2}{T_p}} \cos\left[2\pi\left(f_c + \Delta_i\right) t\right] dt = S_i^I + I_i^I + C_i^I + n_i^I, \tag{16}$$

$$Y_i^Q = \int_0^{T_p} \tilde{R}_i(t) \sqrt{\frac{2}{T_p}} \sin\left[2\pi\left(f_c + \Delta_i\right) t\right] dt = S_i^Q + I_i^Q + C_i^Q + n_i^Q, \tag{17}$$

where $\tilde{R}_i(t)$ is given in (2).

Based on the above equations,

$$|Y_i| = \sqrt{\left(S_i^I + I_i^I + C_i^I + n_i^I\right)^2 + \left(S_i^Q + I_i^Q + C_i^Q + n_i^Q\right)^2}. \tag{18}$$

Define

$$\theta_i \triangleq \pi f_{di} T_p, \tag{19}$$

we could get

$$|Y_i| = \frac{\sqrt{\frac{A_i^2 \sin^2\theta_i}{\theta_i^2} + \sum_{k=1,k\neq i}^{N} \frac{2A_i B_k \sin\theta_i \sin\theta_k \cos(\theta_i - \theta_k)}{[\pi(k-i)+\theta_k]\theta_i}}}{+\left(\sum_{k=1,k\neq i}^{N} \frac{B_k \sin\theta_k \cos\theta_k}{\pi(k-i)+\theta_k}\right)^2 + \left(\sum_{k=1,k\neq i}^{N} \frac{-B_k \sin^2\theta_k}{\pi(k-i)+\theta_k}\right)^2 + M_i^2 + n_i^2}. \tag{20}$$

There are two special cases as follows:

1. If there is no Doppler shift, then $f_{di} = f_{dk} = \theta_i = \theta_k = \sin\theta_i = \sin\theta_k = 0$ and $\frac{\sin^2\theta_i}{\theta_i^2} = 1$, and thus (20) is simplified to

$$|Y_i(t)| = \sqrt{A_i^2 + M_i^2 + n_i^2}. \tag{21}$$

   Because the RSN waveforms provide orthogonality under the circumstances of zero Doppler effect, all interferences among any sensor are eliminated.
2. If there is only one sensor, interferences no longer exist, then (20) becomes

$$|Y_i| = \sqrt{A_i^2 \sin c^2(\theta_i) + M_i^2 + n_i^2}. \tag{22}$$

   From the definition of θ_i (see (19)), we know that if $f_{di} T_p = k$, where $k = \pm 1, \pm 2, \pm 3 \ldots$, then Y_i is composed of only clutter and noise. In this case, the performance of single noncoherent radar is severely terrible. To simplify (20), we define $\xi = E\left\{\frac{\sin\theta_i}{\theta_i}\right\}$, $\Psi = E\left\{\frac{\sin\theta_k \cos\theta_k}{\pi(k-i)+\theta_k}\right\}$, $\omega = E\left\{-\frac{\sin^2\theta_k}{\pi(k-i)+\theta_k}\right\}$. Then

it can be approximated to

$$|Y_i| \cong \left| A_i\xi + \sum_{k=1,k\neq i}^{N} B_k\Psi + \sum_{k=1,k\neq i}^{N} B_k\omega + n_i \right|. \tag{23}$$

$|Y_i|$ approximately follows Rayleigh distribution with parameter

$$\alpha = \sqrt{\gamma^2\xi^2 + (N-1)\rho^2(\Psi^2+\omega^2) + \eta^2 + \sigma^2}. \tag{24}$$

Similarly, we apply the SCA diversity scheme and Eqs. (12)–(15) to analyze the detection performance in noncoherent RSNs.

1.1.4 Performance Analysis of Coherent and Noncoherent RSN

In this section, we analyze the detection performance versus SNR and the detection versus Doppler shift, respectively, for both coherent and noncoherent RSN via Monte Carlo simulations. We assume each $f_{di_{max}}$ is the same for different i. Other parameters are $T_p = 1$ ms and $P_n = P_s$. The mean value and variance of B_k are equal to those of A_i. Clutter-to-noise ratio (CNR) is 6 dB. We perform 10^6 times Monte Carlo simulations.

Figure 2 and Fig. 3 illustrate detection performances for different maximal Doppler shifts when the SNR is fixed. Figure 2 is for coherent RSN, respectively, while Fig. 3 is for noncoherent system respectively.

These figures reveal a general tendency, that is in the same RSN, at the same SNR, the larger Doppler shift, the worse detection performance. The single coherent radar is an exception because the exact Doppler shift is known to the demodulation system, and thus the performance is exact the same in spite of different Doppler shift.

Comparing the subfigures in Fig. 2, we may see that at lower SNR, Doppler uncertainty results in larger variance in performance. When SNR increases to higher value, it would better combat Doppler uncertainty.

In noncoherent RSN system, although it is the same tendency that the larger Doppler shift, the worse detection performance, the variance of performances are much larger than those of coherent system. Also, the degradation of RSN performance is larger than single radar as the Doppler shift increases. This implies that for noncoherent RSN, more radars are needed to combat the Doppler shift ambiguity.

In summary, Sect. 1.1 shows that the proposed RSN can provide much better detection performance than that of a single radar sensor for fluctuating targets, in terms of probability of false alarm and miss detection. Meanwhile, the coherent system is more robust to the noncoherent RSN with the impact of Doppler shift on both coherent and noncoherent RSN detection systems at the presence of clutter, and more details are discussed in [12–16].

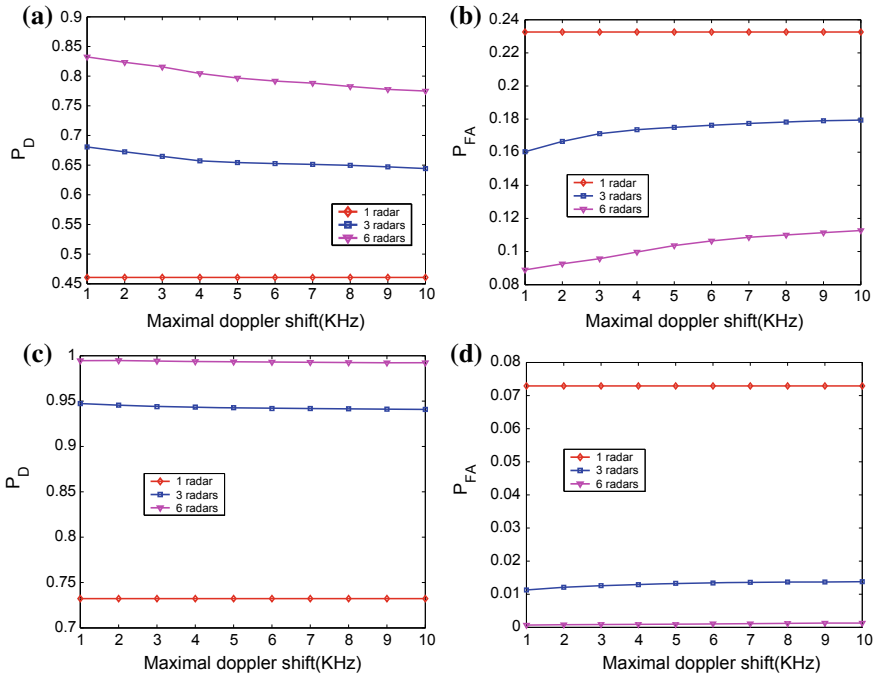


Fig. 2 Performance versus Doppler shift for coherent RSN, **a** probability of detection when SNR = 1 dB, **b** probability of false alarm when SNR = 1 dB, **c** probability of detection when SNR = 10 dB, and **d** probability of false alarm when SNR = 10 dB

1.2 Distributed-RSN and MIMO-RSN in Fading Channels

In this Section, we compare distributed-RSN with multiple-input-multiple-output radar sensor network (MIMO-RSN) in terms of target detection performance. Section 1.1 has demonstrated the advantage of both coherent and noncoherent RSN in target detection. MIMO radar sensor systems have also been a popular research area starting from the beginning of the century due to its excellent performance in target detection compared to traditional radar sensor systems [17–21]. Therefore, we are interested in the question: Is a distributed-RSN system superior to a MIMO-RSN for target detection, or vice versa? The following content will offer some clues.

1.2.1 Channel Models

We model the wireless propagation of RSN under small multi-path fading since RSN sensors are powered by a battery with small ranges of the detection [22, 23].

The model of distributed-RSN is showed in Fig. 4a. ρ_{1i} and ρ_{2i} are the complex channel gain of the signal transmitted and received by sensor i, respectively.

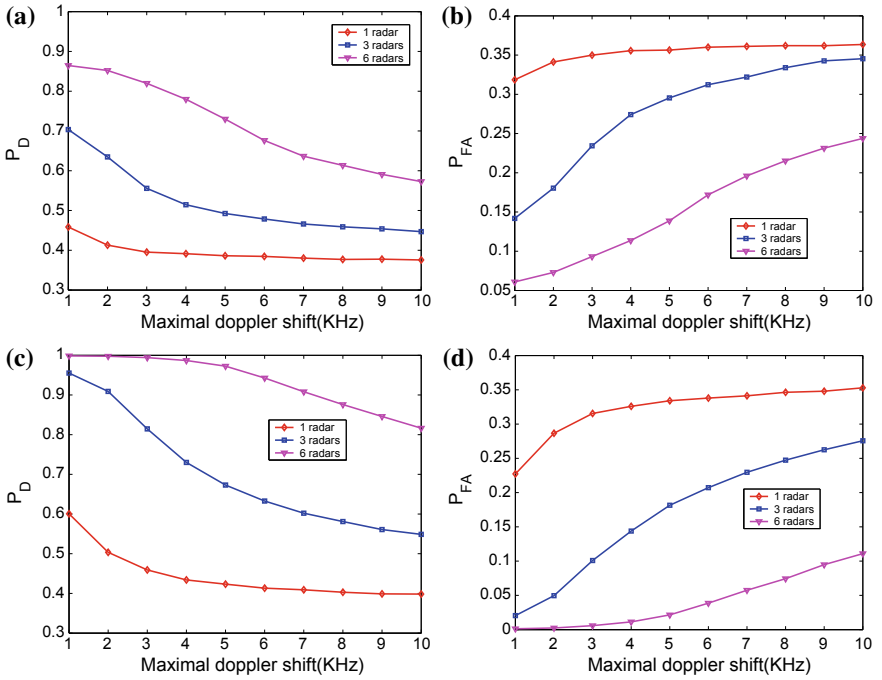


Fig. 3 Performance versus Doppler shift for noncoherent RSN, **a** probability of detection when SNR = 1 dB, **b** probability of false alarm when SNR = 1 dB, **c** probability of detection when SNR = 10 dB, and **d** probability of false alarm when SNR = 10 dB

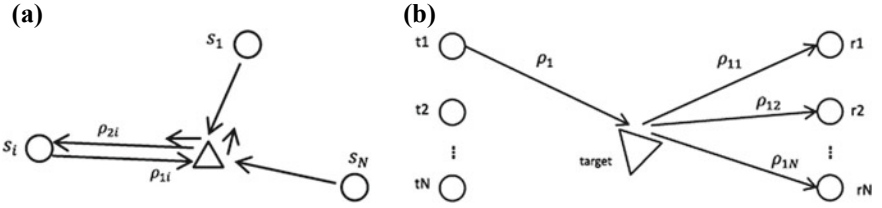


Fig. 4 **a** Distributed-RSN model and **b** MIMO-RSN model

Thus,

$$A_{ri} = G_1 \cdot \alpha \cdot \rho_{1i} \cdot \rho_{2i} \cdot A_{ti}, \tag{25}$$

where A_{ti} and A_{ri} are the transmitted and the received signals of sensor i. G_1 is the product of the transmitting antenna gain and the receiving antenna gain, and α is the RCS parameter. The average received power of sensor i is

$$E\left[A_{ri}^2\right] = E\left\{[\rho_{1i} \cdot \rho_{2i}]^2\right\} \cdot A_{ti}^2 = 4\eta^4 A_{ti}^2, \tag{26}$$

assuming that ρ_{1i} and ρ_{2i} are independent and have the same Rayleigh parameter η.

The model of MIMO-RSN is shown in Fig. 4b. The transmit signal of antenna t_1 obtains complex gain ρ_1 and ρ_{11}, ρ_{12},..., ρ_{1N} before being received by antenna r_1, $r_2, \ldots, r_N$, respectively. The channel gain matrix $\mathbf{H}$ of MIMO-RSN is modeled as

$$\mathbf{H} = G_1\alpha \begin{bmatrix} \rho_1\rho_{11} & \cdots & \rho_1\rho_{1j} & \cdots & \rho_1\rho_{1N} \\ \vdots & & \vdots & & \vdots \\ \rho_i\rho_{i1} & \cdots & \rho_i\rho_{ij} & \cdots & \rho_i\rho_{iN} \\ \vdots & & \vdots & & \vdots \\ \rho_N\rho_{N1} & \cdots & \rho_N\rho_{Nj} & \cdots & \rho_N\rho_{NN} \end{bmatrix}. \tag{27}$$

1.2.2 Signal Combinations and Target Detection

Diversity combining of the independent signal path is an important technique for mitigating the negative effects of signal fading on the wireless propagation, since the randomness of wireless channel induces a power penalty on the performance of RSN systems. For distributed-RSN, we apply maximal ratio combining (MRC) and equal gain combining (EGC) in this book. EGC only applies co-phasing to the signals by multiplication of $\alpha_i = e^{-j\theta_i}$. MRC not only co-phases the signals, but also weight the signals by choosing the $\alpha_i = \frac{r_i}{\sqrt{N_0}}e^{-j\theta_i}$ to maximize the output SNR $\gamma_{\sum}$.

In MIMO-RSN, we may obtain several parallel channels with known channel gain σ_i via singular value decomposition(SVD). Water-filling, equal-power, and channel inversion are frequently used power allocation methods to evaluate the performance of MIMO-RSN. For more details, readers may refer to [24].

At the presence of a target in the sensor detection region, the measurement can be assumed the result of noise only (denoted as I), or the combined result of noise and echoes from a target (denoted as E). The two hypotheses can also be expressed as:

$$\begin{aligned} &(1)\ I : Y = \sum_{i=1}^{N} n_i \\ &(2)\ E : Y = R + \sum_{i=1}^{N} n_i \end{aligned} \tag{28}$$

where Y is the received signal measurement and R is the combination of echoes from a target.

In a distributed-RSN,

$$R = \sum_{i=1}^{N} \alpha_i r_i e^{j\theta_i}, \tag{29}$$

where $\alpha = e^{-j\theta_i}$ for EGC and $\alpha_i = \frac{r_i}{\sqrt{N_0}} e^{-j\theta_i}$ for MRC.

In the MIMO-RSN,

$$R = \sum_{i=1}^{N} \sigma_i x_i, \tag{30}$$

where σ_i is the eigenvalue generated by SVD, and x_i is the signal amplitude on the ith channel. Thus, x_i^2 is the power allocated on the ith channel.

The likelihood ratio test leads to the decision rule:

$$\frac{P_E(y)}{P_I(y)} \underset{\text{no target}}{\overset{\text{target exist}}{\gtrless}} -\lambda \tag{31}$$

where λ is a constant number chosen to get the maximum P_d while P_{fa} does not exceed a limitation.

The noise n_i follows normal distribution. As for a distributed-RSN, it is hard to get the function of P_d, but we can obtain the system performance by determining the threshold of detection:

$$P_{fa} = \frac{1}{2}\left[1 - \text{erf}\left(\frac{T}{\sqrt{2N\beta^2}}\right)\right]. \tag{32}$$

Therefore, threshold $T = \sqrt{2N\beta^2}\text{erf}^{-1}(1 - 2P_{fa})$. A decision that the target is present if $Y > T$ and there is no target vice versa.

In order to get the SNR at the receiver, the relationship between transmit signal amplitude and the average SNR needs to be obtained.

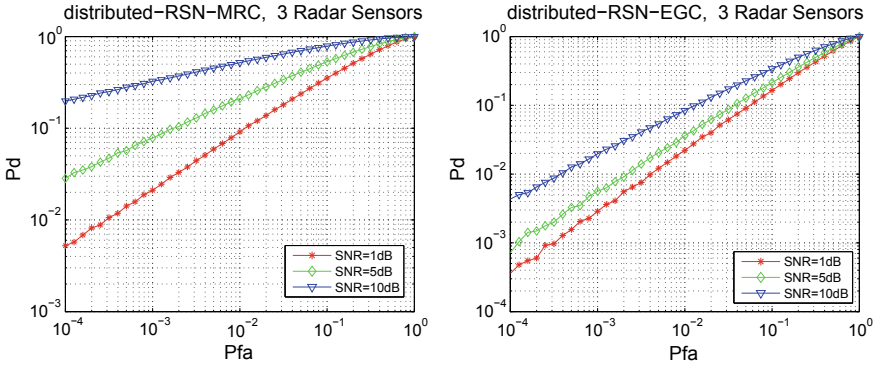
In a distributed-RSN with MRC:

$$A_t = \sqrt{\frac{E\{SNR\} \cdot N_0}{4\eta^4 \cdot N}} \tag{33}$$

In a distributed-RSN with EGC:

$$A_t = \sqrt{\frac{E\{SNR\} \cdot N_0}{\left(4 + \frac{\pi^2}{4}(N-1)\eta^4\right)}} \tag{34}$$

As for MIMO-RSN, we derive the formula of P_d:



(a) MRC (b) EGC

Fig. 5 Detection performance of distributed-RSN

$$P_d = \int_T^{+\infty} P\left(R|E\right) dR = \frac{1}{2}\text{erfc}\left[\text{erfc}^{-1}(2P_{fa}) - \frac{\sum_{i=1}^{N} \sigma_i x_i}{\sqrt{2N\beta^2}}\right], \tag{35}$$

where x_i varies with the power allocation method.

The detection method above is under the assumption of one moving target. The multi-target performance in respect of statistics can also be analyzed. We may assume a RSN knows there are m targets within the range. To make the problem tractable, we assume these m targets are independent, then the probability that all targets can be detected turns out to be P_d^m. Also, the probability that at least one target has been false alarmed is $1 - (1 - P_d)^m$.

1.2.3 Performance Evaluation

The target detection performance of distributed-RSN is plotted in Fig. 5, which shows characteristics of MRC and EGC. The data imply that the increase of SNR has a better improvement in MRC than in EGC, which is because the EGC only utilizes phase information and MRC sets weighting coefficients to take full advantage of good channel.

Figure 6 compares distributed-RSN with MIMO-RSN. It can be seen that there exits a point where they get the same P_d and P_{fa} at the same SNR. MIMO-RSN performs better when P_{fa} is higher than this point, and the distributed-RSN performs better when P_{fa} is lower.

In summary, this section mainly illustrates the channel models of distributed-RSN and MIMO-RSN. The optimal fusion scheme for distributed-RSN is MRC while the optimal power control for MIMO-RSN is water-filling. Distributed-RSN is superior than MIMO-RSN in P_d at both the same SNR and P_{fa}. More details of the performances can be seen in [22, 23].

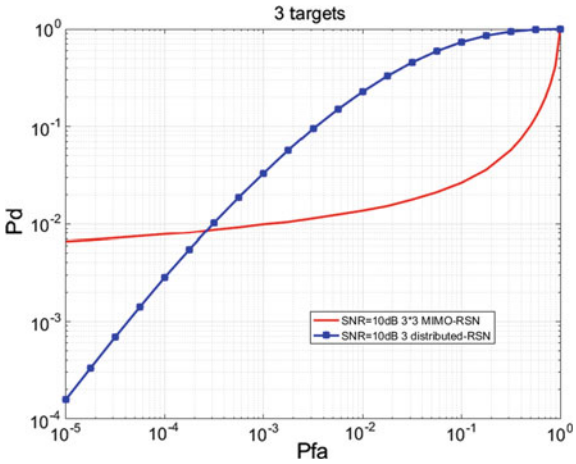


Fig. 6 Detection performance of distributed-RSN and MIMO-RSN

1.3 Nodes Deployment, Clustering Techniques, and Information Fusion

In this section, we investigate the node deployment, clustering, and fusion schemes in WSN. Node deployment is a major challenge to a successful implementation of WSN. The goal of the deployment is to ensure that the WSN can achieve the expected detection performance with high-energy efficiency. Clustering and fusion have also been extensively studied in WSN to extend networks' lifetime, as well as improve energy efficiency. In the following subsection, we shall first analyze two deployment strategies, named Hexagonal Deployment Strategy (HDS) and Diamond Deployment Strategy (DDS), respectively, combined with two multi-hop decision fusion rules: binary transmission (BT) and no binary transmission (NBT) [25]. Then two fuzzy clustering schemes with a graphical optimal routing selection (GORS) algorithm for multi-hop WSN [26] will be presented.

1.3.1 Node Deployment Strategies and Multi-hop Fusion

Hexagonal Deployment Strategy (HDS)

The HDS is a strategy that place finite sensors to form mutually mosaic hexagons in the monitoring plane area. The following processes are taken.

1. Take the vertex K of bottom left margin of this rectangle as a starting point and R as the length of each segment, dividing equally the rectangle hemline. Starting with these equal division points, make rays with 60° and 120° until they intersect with the boundary of this rectangle.

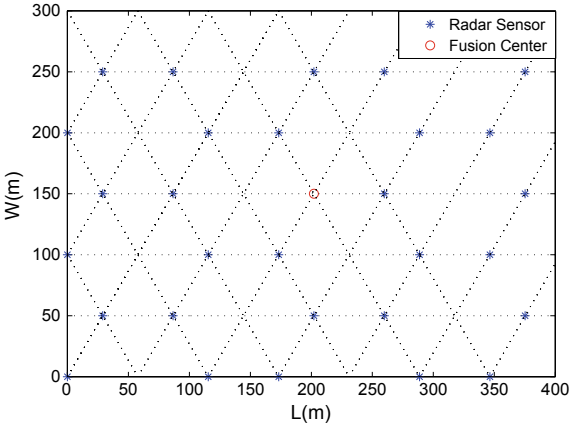


Fig. 7 Array N = 30 sensors in the area using the HDS strategy

2. Take K as a starting node and $\frac{\sqrt{3}}{2}R$ as the length of each segment, dividing equally the rectangle left boundary. Starting with these equal division nodes, make rays with 60° and rays paralleling to the hemline of this rectangle until they intersect with the boundary of this rectangle.
3. We need to select the proper points to place radar sensors there to form hexagons. First, pick out the points in the rectangle from these crossing points obtained according to above process. Second, in each odd row, respectively, taking the first point as the starting point, pick out every third point, and then taking the third point as the starting point, pick out every third point. Third, in each even row, respectively, taking the first point as the staring point, pick out every third point and then taking the second point as the starting point, pick out every third point.
4. Select the node which is closest to the geometric center of this rectangle to place a fusion center of WSN from nodes composing these hexagons and nodes in the center of these hexagons.

The number of sensors in the WSNs monitoring rectangular area for HDS is $N_1 \approx \frac{2}{3} \times \left[\frac{L}{R}\right] \times \left[\frac{W}{\frac{\sqrt{3}}{2}R}\right]$, where N_1 is a approximate number of sensors rather than an accurate one, $[*]$ is an integer which is not less than $*$. After the above placing process, a whole nodes placement diagram is completed. Here, we use a $400 \times 300\,\mathrm{m}^2$ area (notice that the square is a special case of rectangle) as an example to exhibit the HDS. Figure 7 shows the graphical deployment of $N_1 = 30$ nodes in a $400 \times 300\,\mathrm{m}^2$ area using HDS.

Diamond Deployment Strategy (DDS)

The DDS is a strategy that places finite radar sensors to form mutually mosaic diamonds in the monitoring plane area. The following processes are used to deploy sensors in DDS.

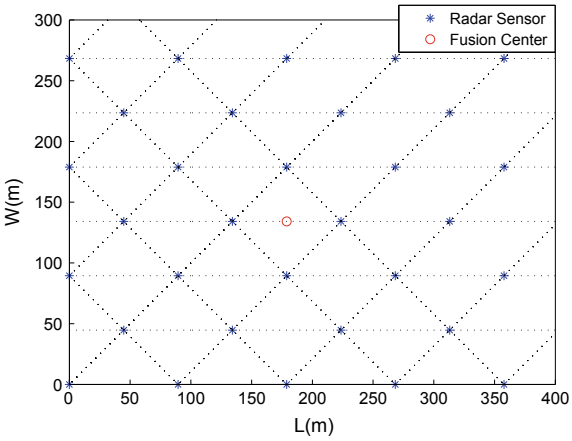


Fig. 8 Array N = 30 sensors in the area using the DDS strategy

1. Take the vertex K of bottom left margin of this rectangle as a starting point and R as the length of each segment, dividing equally the rectangle bottom boundary. Starting with these equal division points, make rays with 45° and 135° until they intersect with the boundary of this rectangle.
2. Take K as a starting point and $\frac{\sqrt{3}}{2}R$ as the length of each segment, dividing equally the rectangle left boundary. Starting with these equal division points, make rays with 45° and rays paralleling to the hemline of this rectangle until they intersect with boundary of this rectangle.
3. Select the nodes in the rectangle from these crossing nodes to place radar sensors there to form diamonds.
4. Select the node which is closest to the geometric center of this rectangle to place a fusion center of WSN from nodes composing these diamonds and nodes in the center of these diamonds.

The number of sensors in the WSNs monitoring rectangular area for DDS is $N_2 \approx \left[\frac{L}{R}\right] \times \left[\frac{W}{\frac{R}{2}}\right]$, where the N_2 is not an accurate one, but an approximate number of sensors instead. After the above placing process, a whole array nodes diagram is completed. Here, we use a $400 \times 300\,\mathrm{m}^2$ area as example to exhibit the DDS strategy. Figure 8 shows the graphical deployment of $N_2 = 30$ nodes in a $400 \times 300\mathrm{m}^2$ area using DDS.

Decision Fusion Rules with Binary Transmission (BT)

In this decision fusion rule, each relay node tries to retrieve the decision sent from its source node in spite of fading and noise distortion. These relay nodes make a binary decision when receiving signals. Assume all the channels are independent of each other and each of them can be modeled as a path-loss channel. Noise in all channels are Gaussian with zero mean and variance σ^2 and are independent of each other.

The signal amplitude that every radar sensor send for detection is A_a and for relay is A_b. Assume M_k denotes the number of relay nodes between the kth local radar

sensor and the fusion center, with $k = 1, 2, \ldots, N-1$. The h_k^i is the corresponding channel gain and $i = 0, 1, \ldots, M_k$ is the hop index. The process of target detection in BT is described below.

1. Every sensor sends a signal with amplitude A_a out for detection and receives an echo from the target, independently.
2. According to the echoes, each sensor individually makes a binary decision (local decision) : $u_k^0 = +1$ is made if H_1 is decided, and $u_k^0 = -1$ is made otherwise. They each send the signal $v_k^0 = A_b u_k^0$ out for relay.
3. Local decisions made at $N-1$ sensors are transmitted over path-loss fading channels to the fusion center through several relay nodes. Every relay node makes a binary decision which u_k^i is either $+1$ or -1 and sends signal v_k^i out. There

$$u_k^i = sign(v_k^{i-1} h_k^{i-1} + n_k^{i-1}) \tag{36}$$

$$v_k^i = A_b u_k^i \tag{37}$$

4. The decisions are sent to the fusion center, finally. Let y_k denotes the input signal of the fusion center from the kth sensors, thus,

$$y_k = A_b u_k^{M_k} h_k^{M_k} + n_k^{M_k} \tag{38}$$

When $u_k^{M_k}$ is determined, y_k obey the Gaussian distribution with mean $A_b u_k^{M_k} h_k^{M_k}$ and variance σ^2. Based on the received data y_k for all sensors, the fusion center decides whether having a target or not.

Define $P_{dk}^{(c)}$ and $P_{fk}^{(c)}$ as the probability of detection and probability of false alarm, respectively, at the last relay.

$$P_{dk}^{(c)} = P(u_k^{M_k} = +1 | H_1) \tag{39}$$

$$P_{fk}^{(c)} = P(u_k^{M_k} = +1 | H_0) \tag{40}$$

They are different from the local performance indices P_{dk} and P_{fk}.

The optimal likelihood ratio (LR)-based fusion statistics for the multi-hop systems with BT is denoted by Λ_1. Given $P_{dk}^{(c)}$ and $P_{fk}^{(c)}$, the LR with the fusion statistic can be written as

$$\Lambda_1 = \frac{f(Y|H_1)}{f(Y|H_0)} = \prod_{k=1}^{N-1} \frac{f(y_k|H_1)}{f(y_k|H_0)} = \prod_{k=1}^{N-1} \frac{P_{dk}^{(c)} + (1 - P_{dk}^{(c)}) e^{-(2y_k h_k^{M_k} A_b)/\sigma^2}}{P_{fk}^{(c)} + (1 - P_{fk}^{(c)}) e^{-(2y_k h_k^{M_k} A_b)/\sigma^2}} \tag{41}$$

Decision Fusion Rules with No Binary Transmission (NBT)

In this fusion rule, we assume that relay nodes do not make binary decision when transmitting data. They simply forward the information from source nodes to the

fusion center. Other conditions are the same as BT case. The process of target detection in NBT is described as follows.

1. The first two steps are the same as the fusion rule of BT.
2. Local decisions made at $N-1$ sensors are transmitted over path-loss fading channels to the fusion center through several relay nodes. Every relay node simply forwards the information v_{2k}^i from source sensor.

$$v_{2k}^i = v_{2k}^{i-1} h_k^{i-1} + n_k^{i-1} \tag{42}$$

3. The decisions are sent to the fusion center, finally. The input signal of the fusion center from the kth sensor is

$$y_{2k} = h_k^{M_k} v_{2k}^{M_k} + n_k^{M_k} = h_k^{M_k}[\ldots h_k^2(h_k^1(h_k^0 v_{2k}^0 + n_k^0) + n_k^1) + n_k^2 \ldots] + n_k^{M_k} \tag{43}$$

On account of that n_k^i is additive Gaussian noise, when the u_k^0 is fixed, y_{2k} obeys the Gaussian distribution. We set $\mu_k = h_k^{M_k} \ldots h_k^2 h_k^1 h_k^0 A_b$ and $\sigma_k^2 = [(h_k^{M_k} \ldots h_k^2 h_k^1)^2 + (h_k^{M_k} \ldots h_k^2)^2 + \cdots + (h_k^{M_k})^2 + (1)^2]\sigma^2$. When $u_k^0 = 1$ at the kth local sensor, y_{2k} obeys the Gaussian distribution with mean μ_k and variance σ_k^2 and when $u_k^0 = 0$ at the kth local sensor, y_{2k} obeys the Gaussian distribution with mean $-\mu_k$ and variance σ_k^2.

Due to (43), $f(y_{2k}|H_1)$ and $f(y_{2k}|H_0)$ are decomposed into

$$f(y_{2k}|H_1) = P_{dk} f(y_{2k}|u_k^0 = 1, H_1) + (1 - P_{dk}) f(y_{2k}|u_k^0 = -1, H_1) \tag{44}$$

$$f(y_{2k}|H_0) = P_{fk} f(y_{2k}|u_k^0 = 1, H_0) + (1 - P_{fk}) f(y_{2k}|u_k^0 = -1, H_0) \tag{45}$$

Therefore, in NBT situation, the LR can be written as

$$\Lambda_2 = \prod_{k=1}^{N-1} \frac{P_{dk} + (1 - P_{dk}) e^{-(2y_{2k}\mu_k)/\sigma_k^2}}{P_{fk} + (1 - P_{fk}) e^{-(2y_{2k}\mu_k)/\sigma_k^2}} \tag{46}$$

Performance Evaluation

Figure 9 shows the performances of HDS, DDS, and random deployment strategy (RDS) in terms of probability of detection with BT and NBT decision fusion rules. Three conclusions can be drawn here. First, the performance for BT fusion rules is better than that for NBT fusion rules. Second, both HDS and DDS are better than RDS in terms of detection probability. Third, DDS is superior to HDS no matter in BT or NBT. More reference for sensor deployment and fusion schemes can be found in [27–38].

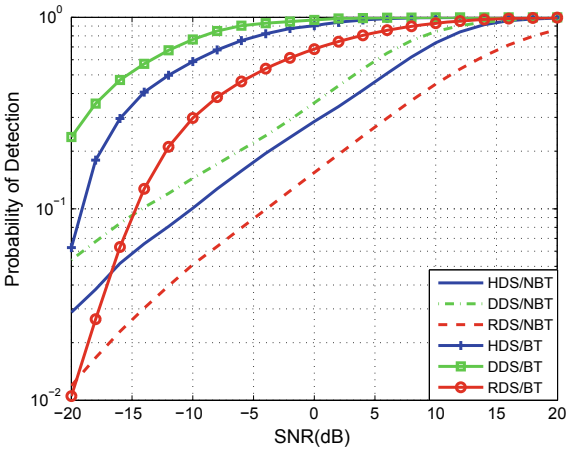


Fig. 9 Probability of detection versus channel SNR using HDS/DDS/RDS and BT/NBT for pass-loss fading channels, N = 20

1.3.2 Fuzzy Clustering and Multi-hop Fusion

For WSN with a large number of sensors, the data set received by the system terminal is huge. These data contain a lot of redundancy. Information fusion technology can reduce the redundancy and improve the accuracy of the information. However, the data transmission bandwidth is limited, and there is a collection of errors or distortion of the data received by the sensors.

Clustering algorithms can effectively reduce the energy consumption of wireless networks, prolong the network life cycle, enhance the network coherence, and reduce the development of data [39–41]. Its basic idea is dividing the wireless sensor network into different areas, called clusters. And then, set the central control node in the clusters, which is the CH.

The energy is primarily consumed for detection and data transmission. We implement the free space and the multi-path fading channel models. The transmitter dissipates energy to run the radio electronics and the power amplifier, and the receiver dissipates energy to run the radio electronics.

Fuzzy Clustering

Fuzzy clustering has been an efficient tool for data science. This section presents two fuzzy clustering schemes in WSN data processing for target detection. Small-scale fading is considered in fuzzy logic system (FLS) design (FLS with three-antecedents, F3) to compute the likelihood to be a CH for each sensor at the first stage. In case of single-hop routing, fuzzy c-means with singular value decomposition-QR (FCMSVDQR) approach is proposed to decide the final CH. As for multi-hop routing, firstly the sensor with the highest FLS likelihood will be elected a CH. Secondly, a graphical optimal routing selection (GORS) algorithm is applied for multi-hop data transmission.

FCMSVDQR: NCHs directly transmit the decision results to the corresponding CH without any relay. In order to improve the detection performance of CHs,

FCMSVDQR approach is proposed to select the optimal CHs based on the likelihood of a sensor being elected to be a CH, which is derived from F3.

Step 1: Using FCM clustering to choose temporary CHs based on the output of F3. Assume the likelihood of a sensor being elected to be a CH of N sensors in a cluster is $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, we use FCM to part the vector $\mathbf{x}$ to M clusters, and choose the cluster with the largest center as the temporary CH cluster. M is an integer and can be computed by

$$M = floor(C_{prob} * N) \tag{47}$$

where C_{prob} (say 30%) is a constant ratio.

Step 2: Suppose $h_{i,j}$ is the channel gain from the ith sensor to the jth sensor in a cluster with N sensors, where $1 \leq i, j \leq N$ and $i \neq j$, the vector $\mathbf{h_j}$ is expressed as $\mathbf{h_j} = (h_{1,j}, h_{2,j}, \ldots, h_{i,j}, \ldots, h_{N,j})^T$, and the channel gain matrix $H_{(N-1)\times N}$ is

$$H = (\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_j}, \ldots, \mathbf{h_N}) \tag{48}$$

Assume $N_t (N_t > 1)$ is the number of temporary CHs elected by FCM and the index set of temporary CHs is the vector $\mathbf{y} = (y_1, y_2, \ldots, y_{N_t})$, the channel gain matrix of temporary CHs is $H_t = H(:, \mathbf{y})$ and $r = rank(H_t) \leq N_t$.

$$H_t = (\mathbf{h_{y_1}}, \mathbf{h_{y_2}}, \ldots, \mathbf{h_{y_{N_t}}}) \tag{49}$$

Step 3: Calculate the SVD of H_t,

$$H_t = U \Sigma V^T \tag{50}$$

We choose σ_1 and $V = (\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}, \ldots, \mathbf{v_{N_t}})$, where σ_1 is the largest singular value of H_t and $\mathbf{v_k}$ is the kth vector of V.

Step 4: Using QR decomposition with column pivoting, determine E such that

$$\mathbf{v_1}^T E = QR \tag{51}$$

where Q is a unitary matrix; and E is the permutation matrix. The position of 1 in the first column of E corresponds to the final CH.

GORS: NCHs transmit the decision results through multi-relay to the corresponding CH. Here we choose the sensor with the largest output of FLS as the CH in a cluster. The GORS algorithm is proposed to utilize the highest fading path from each NCH to the CH in a cluster. The input is a completed graph: associated with each directed edge (n_i, n_j) is the fading $h_{i,j}$ to traverse arc from the ith node to the jth node. The pseudocode for GORS algorithm is shown in Algorithms 1 and 2 (more details are shown in [26]).

Algorithm 1 Initialize for GORS

1: $n\_cur = cluster\ head$
2: Struct Node n_i
{
$n_i.known = false$
$n_i.next = cluster\ head$
$n_i.mark = i$
$n_i.fading = h_{i,cur}$
}
3: Vector $\mathbf{V_{node}} \longleftarrow \{n_i \in \mathbf{V_{node}}\}$
4: Sort the elements of $\mathbf{V_{node}}$ based on $n_i.fading$ in descending order

Algorithm 2 Repeat Phase for GORS

1: **while** existing unknown element in $\mathbf{V_{node}}$ **do**
2: $\mathbf{V_{node}}[1].known = TRUE$
3: $n\_cur = \mathbf{V_{node}}[1]$
4: Put $\mathbf{V_{node}}[1]$ in the last position of $\mathbf{V_{node}}$
5: Vector $\mathbf{V_{selectNode}} \longleftarrow \{n_j \in \mathbf{V_{node}} \&\& h_{j,cur} > n_j.fading\}$
6: **if** (**then**$\mathbf{V_{selectNode}}$ is empty)
7: continue
8: **end if**
9: Erase the element of $\mathbf{V_{selectNode}}$ from $\mathbf{V_{node}}$
10: **for** each element n_j in $\mathbf{V_{selectNode}}$ **do**
11: $n_j.next = n\_cur.mark$
12: $n_j.fading = h_{j,cur}$
13: **end for**
14: Sort the elements of $\mathbf{V_{selectNode}}$ based on $n_j.fading$ in descending order
15: $\mathbf{V_{node}}$=[$\mathbf{V_{selectNode}}$, $\mathbf{V_{node}}$]
16: Clear $\mathbf{V_{selectNode}}$
17: **end while**

Multi-hop Fusion

Let $M_{k,i}$ denote the number of total hops between the kth sensor and the CH within the ith cluster. Particularly, if $M_{k,i} = 0$, the data from kth NCH are directly transmitted to the ith CH. $u_{k,i}^0$ is the detection decision of the kth sensor. If $1 \leq M_{k,i} \leq N_i - 1$, $u_{k,i}^m$ ($1 \leq m \leq M_{k,i}$) denotes the retrieved decisions made by the mth relay node, where m is the hop index. $y_{k,i}^{m-1}$ means the input signal of the mth relay node corresponding to the kth sensor. Thus,

$$y_{k,i}^m = u_{k,i}^m h_{k,i}^m + n_{k,i}^m, \quad 0 \leq m \leq M_{k,i} \tag{52}$$

where $h_{k,i}^m$ is the fading channel. And $n_{k,i}^m$ is additive Gaussian noise.

Let $P_{dM_{k,i}}^{(r)}$ and $P_{fM_{k,i}}^{(r)}$ denote the detection and false alarm rate of the last relay nodes, for $1 \leq m \leq M_{k,i}$,

$$P_{dm}^{(r)} = P[u_{k,i}^m = +1|H_1], \tag{53}$$

$$P_{fm}^{(r)} = P[u_{k,i}^m = +1|H_0]. \tag{54}$$

Suppose $T_{k,i}^m$ is the decision threshold of mth relay node to achieve the constant false alarm rate (CFAR). Hence, given that the noise is Gaussian, we have

$$u_{k,i}^m = \begin{cases} +1, & y_{k,i}^m > T_{k,i}^m, \\ -1, & y_{k,i}^m \le T_{k,i}^m. \end{cases} \tag{55}$$

Therefore, in the low SNR case, the multi-hop fusion statistic of CHs can be expressed as

$$\Lambda_a = \sum_{k=1, M_{k,i}>0}^{N_i} (P_{dM_{k,i}}^{(r)} - P_{fM_{k,i}}^{(r)}) h_{k,i}^{M_{k,i}} y_{k,i}^{M_{k,i}} + \sum_{k=1, M_{k,i}=0}^{N_i} (P_{dk} - P_f^{(l)}) h_{k,i}^0 y_{k,i}^0. \tag{56}$$

Our goal is to derive $P_{dM_{k,i}}^{(r)}$ and $T_{k,i}^m$ using prior information. Here we define

$$C_{k,i}^m = P[u_{k,i}^m = +1|u_{k,i}^{m-1} = +1], \qquad \bar{C}_{k,i}^m = P[u_{k,i}^m = +1|u_{k,i}^0 = +1], \tag{57}$$

$$D_{k,i}^m = P[u_{k,i}^m = +1|u_{k,i}^{m-1} = -1], \qquad \bar{D}_{k,i}^m = P[u_{k,i}^m = +1|u_{k,i}^0 = -1]. \tag{58}$$

Assume $P_f^{(r)}$ denotes the CFAR of relay nodes, if $m \ge 2$,

$$\begin{aligned} P_{fm}^{(r)} &= \sum_{u_{k,i}^{m-1}} P[u_{k,i}^m = +1|u_{k,i}^{m-1}, H_0] P[u_{k,i}^{m-1}|H_0] \\ &= P_f^{(r)} Q\left(\frac{T_{k,i}^m - h_{k,i}^{m-1}}{\sigma}\right) + (1 - P_f^{(r)}) Q\left(\frac{T_{k,i}^m + h_{k,i}^{m-1}}{\sigma}\right). \end{aligned} \tag{59}$$

Given $P_f^{(r)}$ and $P_f^{(l)}$, we can derive the $T_{k,i}^m$ using formula (59). Thus, $P_{dm}^{(r)}$ can be computed by

$$\begin{aligned} P_{dM_{k,i}}^{(r)} &= P[u_{k,i}^{M_{k,i}} = +1|H_1] = \sum_{u_{k,i}^0} P[u_{k,i}^{M_{k,i}} = +1|u_{k,i}^0, H_1] P[u_{k,i}^0|H_1] \\ &= P_{dk} \bar{C}_{k,i}^{M_{k,i}} + (1 - P_{dk}) \bar{D}_{k,i}^{M_{k,i}} \end{aligned} \tag{60}$$

Performance Evaluation

This section presents how different cluster numbers influence the clustering performances of the proposed schemes, F3&FCMSVDQR and F3&GORS. Figure 10 illustrates the network's energy consumption of F3&FCMSVDQR and F3&GORS at different cluster number by using the number of round when the first sensor dies.

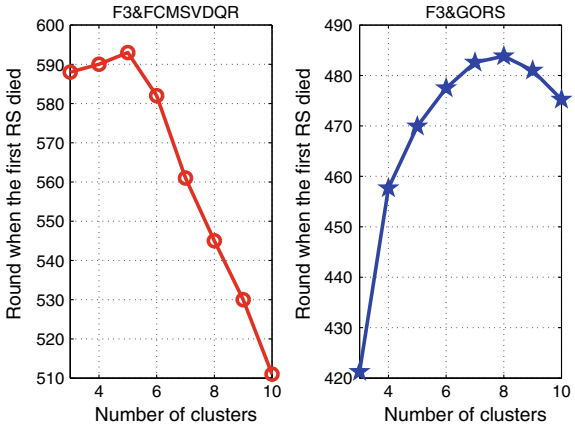


Fig. 10 The number of round when the first sensor dies in WSN in F3&FCMSVDQR and F3&GORS versus different cluster number

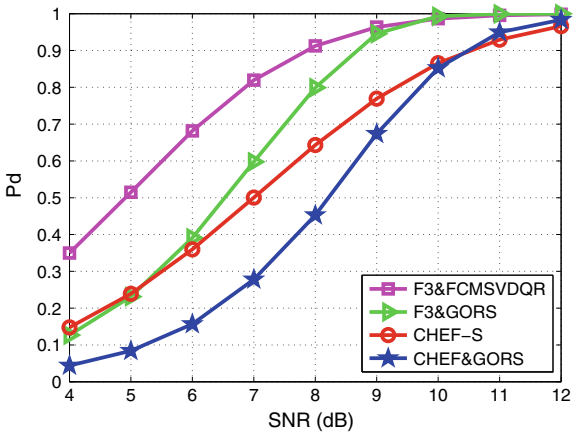


Fig. 11 The P_d of the WSN in F3&FCMSVDQR, F3&GORS, CHEF-S, CHEF&GORS at fixed CFAR (10^{-3}) and cluster number versus different SNRs

Figure 11 presents the P_d of the whole WSN in F3&FCMSVDQR, F3&GORS, CHEF-S, and CHEF&GORS at different SNRs. The ROC curves are generated using 10^6 Monte Carlo runs. From this figure, we can obtain that:

(1) F3&FCMSVDQR provides the robust detection performances, while F3&GORS takes the second place in improving the detection performances at moderate-to-high SNRs.
(2) The P_d of F3&FCMSVDQR and F3&GORS are higher than that of CHEF-S and CHEF&GORS, respectively.

In summary, this section presented two deployment strategies, namely HDS and DDS to deploy finite sensors to achieve a higher expected detection probability with low energy consumption to satisfy the target detection performance in WSN. Under same channel SNR, the DDS achieves highest probability of detection, the HDS gets a lower one, and the random deployment strategy (RDS) is lowest no matter in

BT and NBT. Also, two fuzzy clustering schemes in WSN for target detection are presented, F3&FCMSVDQR and F3&GORS. F3&FCMSVDQR provides the robust detection performances and offers the longest lifetime at large amount of residual alive sensors, while F3&GORS takes the second place in improving the detection performances at moderate-to-high SNRs and the energy efficiency at small amount of residual alive sensors. More references in WSN fuzzy clustering and routing can be seen in [42–49].

2 Node Localization

Apart from target detection, node localization has also been a popular research area in WSN. Node localization means that sensor nodes obtain their own position information in various environments. Localization is of great importance in many applications, such as data routing, target tracking and environment monitoring, and the like. In theory, each wireless sensor can be equipped with a global positioning system (GPS) receiver. However, it is impractical for each sensor to contain a GPS due to many factors including cost, size, and energy efficiency. Therefore, a number of algorithms independent from GPS have been proposed. They are mainly divided into two categories, range-based algorithms and range-free algorithms, as illustrated in Fig. 12.

Range-based algorithms, as shown in its name, are the algorithms based on ranging techniques. Firstly, it needs some beacon nodes aware of their location information accurately. Then, other nodes estimate their distances to these beacon nodes so that their positions can be computed by ranging algorithms, such as trilateration. The implementation of these algorithms will be discussed in this section. Although range-based algorithms can provide high localization accuracy, the cost including the communication and the computation cost can be huge. Moreover, they also have high requirements on hardware.

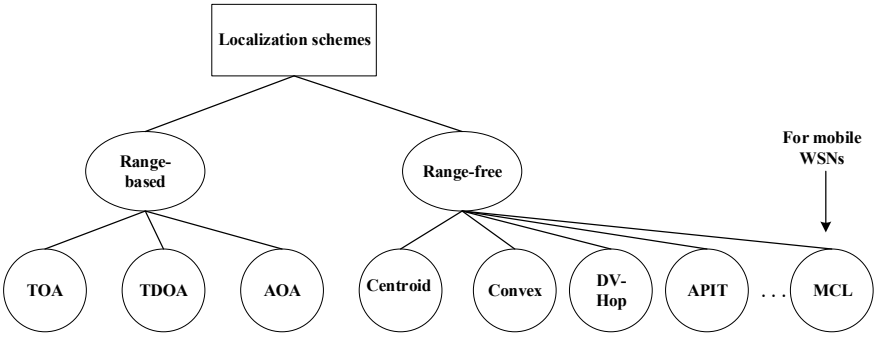


Fig. 12 Localization overview

As a consequence, range-free algorithms are more popular for their low power consumption and low cost in recent years. Different from ranging, range-free algorithms mainly utilize the connectivity information between nodes to estimate the position information. They also need some beacon nodes whose positions are known as the references. Although at present they have not achieved the same level of localization accuracy as that of range-based algorithms, in general, their performances are acceptable for common WSN applications.

2.1 Range-Based Algorithms

In range-based algorithms, the accuracy of distance and angle measurements are critical. Notice that the localization accuracy varies with different ranging scheme. The performance can be influenced by the conditions of wireless channels. There are three typical range-based localization methods that are used most frequently. They are time of arrival (TOA), time difference of arrival (TDOA), and angle of arrival (AOA).

2.1.1 Time of Arrival (TOA)

Time of arrival techniques [50] calculate the distance between two nodes by the measurements of transmit and receive times of signals. It is shown in Fig. 13, where S is the sensor whose position is to be determined, and B_1, B_2, B_3 are beacon nodes. They are all synchronized. Assuming that B_1, B_2, B_3 all send signals at time t_0, and S receives signals at time t_i. The distance between S to B_1, B_2, B_3 can be calculated as $r_i = (t_i - t_0)c$, $i = 1, 2, 3$, respectively, and c is the speed of light. Then, the position of S can be estimated by the trilateration algorithm, which is the intersection of the range circles in Fig. 13

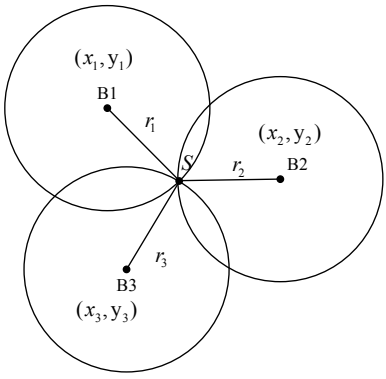


Fig. 13 TOA algorithm

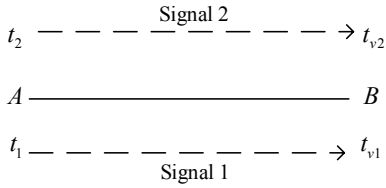


Fig. 14 TDOA algorithm (adapted from [51])

TOA needs very accurate hardware to measure the receive time t_i. Because even very small deviation of time can lead to a large distance error due to the high speed of RF signals. TOA may be acceptable for environments where the speed of RF signal is low.

2.1.2 Time Difference of Arrival (TDOA)

Time difference of arrival algorithm [51] utilizes the difference of the times when separate signals are received to calculate the distance between nodes. A typical approach named multi-signal TDOA is shown in Fig. 14.

At first, node A sends an ultrasound signal with a speed of v_1 at t_1, and node B receives the signal at time t_{v1}. After a while, A sends a RF signal again and its speed is v_2(larger than v_1). The transmitting and receiving time of the RF signal are t_2 and t_{v2}, respectively. The distance of A and B can be calculated as the formula below:

$$d_{AB} = (v_2 - v_1)[(t_{v1} - t_1) - (t_{v2} - t_2)] \tag{61}$$

TDOA is widely used in practice. Since TDOA uses the time discrepancy of different signals to estimate the distance, time synchronization is not needed. TDOA can tolerate small errors of time measurements, and therefore has less requirements for the hardware. Furthermore, the TDOA algorithm can achieve high accuracy in line-of-sight (LOS) conditions. As for NLOS conditions, the performance may degrade significantly. That is the challenge to TDOA.

2.1.3 Angle of Arrival

The angle of arrival (AOA) [52] algorithm locates the nodes by the angle measurements of received signals. In order to obtain the angles of arrived signals from beacon nodes, each node uses two receivers placed at specific distances. In Fig. 15, A is a node whose position is to be estimated. B is a beacon node. A_1 and A_2 are the two receivers. Their distances to A are both $L/2$. x_1 and x_2 can be easily calculated by some ranging techniques. Then, the angle of arrived signals θ can be inferred by x_1, x_2, and L.

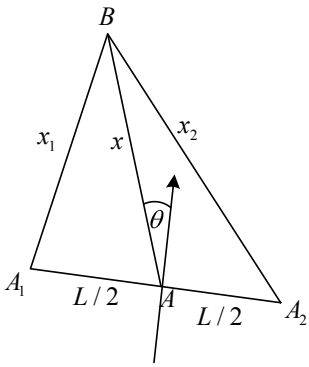


Fig. 15 Basic principle to obtain the angle of arrival (adapted from [52])

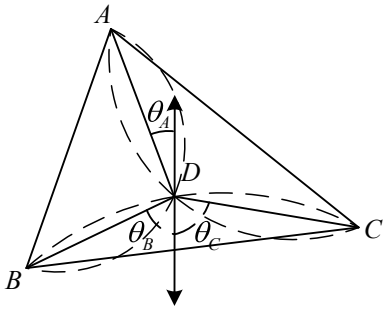


Fig. 16 The AOA algorithm (adapted from [52])

Knowing how to get the angles of arrived signals, Fig. 16 shows the AOA localization, A, B, C are beacon nodes, D is a sensor node whose position is to be estimated. $\angle BDC$, $\angle CDA$, and $\angle ADB$ can be easily calculated by the angles of arrived signals θ_A, θ_B, θ_C. Then the position of node D can be obtained using the triangulation scheme. The triangulation scheme tries to find three circles determined by the beacon nodes and the angle measurements (illustrated by the dashed curves in Fig. 16). The intersecting point of the three circles is the estimated position of node D.

The localization accuracy of AOA algorithm largely depends on the angle measurements. As a consequence, it needs complex antenna arrays which may introduce a high communication cost. Moreover, the performance of AOA algorithm is also influenced by the multi-path and NLOS conditions just as the TDOA algorithm.

2.2 Range-Free Algorithms

Acknowledging that range-based algorithms require huge cost of hardware, e.g., TOA needs high bandwidth and AOA needs many antennas, they may be inappropriate to be used in some applications of WSN. Range-free algorithms are an alternative to achieve accurate node localization. They are characterized by their low commu-

nication cost and computation cost with more nodes needed. In this part, firstly four range-free algorithms are presented: centroid, convex optimization localization, DV-Hop, and approximate point-in-triangular test (APIT). They all show satisfying performances in static WSNs (all sensors remain at the original position after deployment). However, the localization accuracy may degrade significantly once the nodes can move after the deployment. In applications of moving sensors, Monte Carlo localization (MCL) is a promising technique for node localization. It will be described in detail after APIT algorithm. There are a lot of subbranches based on MCL, such as Monte Carlo box (MCB), weighted MCL (WMCL). Both of them can improve the performance of MCL in terms of sampling efficiency and beacon density requirement. They will also be introduced along with MCL in the end of this section.

2.2.1 Centroid Algorithm

The centroid algorithm [53] introduces two idealized radio assumptions which are simple and useful:

- The same transmitted power (distance r) of each node.
- Each node incorporates isotropic antenna.

Firstly, there are some beacon nodes with overlapping communication regions in the network. They are denoted as B_1 to B_n. Their positions are (X_1, Y_1) to (X_n, Y_n) and they form a regular grid. After deployment, the beacon nodes send radio frequency (RF) signals periodically. Each sensor collects the beacon signals that it receives during a time period t, which is predefined as:

$$t = (S + 1 - \varepsilon)T \quad (0 < \varepsilon \ll 1) \tag{62}$$

where S is the sample size of signals sent by each beacon. T is the interval between two consecutive signals. The information of a beacon node B_i is called a connectivity metric CM_i, represented as:

$$CM_i = \frac{N_{recv}(i, t)}{N_{sent}(i, t)} \tag{63}$$

where $N_{sent}(i, t)$ represents the number of beacon signals sent by B_i during the time period t, and $N_{recv}(i, t)$ are the received number of beacon signals sent by B_i within the same period. The sensor node infers a collection of beacon nodes in which each beacon has a connectivity metric exceeding a threshold CM_{thresh}. The threshold is a constant that is predefined (90% is appropriate). Therefore, the collection of beacon nodes is denoted as $B_{i1}, B_{i2}, \ldots, B_{ik}$. Finally, the sensor node that receives the beacon signals determines its position (X_{est}, Y_{est}) as the centroid of these beacon nodes:

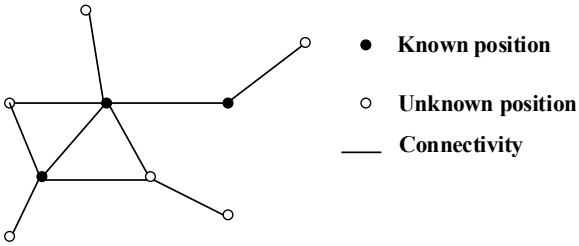


Fig. 17 Graph illustrations of constraints (adapted from [54])

$$X_{est} = \frac{X_{i1} + \cdots + X_{ik}}{k} \tag{64}$$

$$Y_{est} = \frac{Y_{i1} + \cdots + Y_{ik}}{k} \tag{65}$$

The centroid algorithm is simple and needs only low-cost devices. It utilizes the proximity to localize the sensor nodes with unknown positions. Even so, it can achieve promising performance as is shown by experimental results. In outdoor environments, the estimated positions of sensors can be nearly 90% as the reality. Moreover, the centroid algorithm is also suitable for the large distributed networks.

2.2.2 Convex Optimization Localization

The convex optimization localization algorithm [54] is based on connectivity-induced constraints. It collects the communication information of neighbor nodes as the geometric constraints to estimate the unknown positions of sensor nodes in networks. It assumes that a proximity constraint exists when one node can communicate with another. Geometrically, the sensor nodes are within the radio range if they can communicate with each other. The convex optimization algorithm is illustrated in Fig. 17

- **Given**: positions of some beacon nodes (black dots in the figure)
- **Find**: the possible positions of the sensor nodes (the hollow dots in the figure)
- **Subject to**: the proximity constraints inferred from the communication information.

Since the positions of nodes are constrained by the radio range of neighbor beacon nodes, all the constraints together show that the feasible positions are the intersection of the constrained regions, as illustrated in Fig. 18. The shadow area stands for the potential position set of the hollow node, restricted by its neighboring beacon nodes. (The radio coverage area of each node is assumed to be a circle.) It is seen from Fig. 18 that the shaded area becomes smaller as more neighbor beacon nodes involved. As a result, the estimation accuracy will be higher. Finally, once the shaded area is small enough to estimate the position, it is bounded by a minimal rectangle. The center of the minimal rectangle is chosen as the position of the hollow dot.

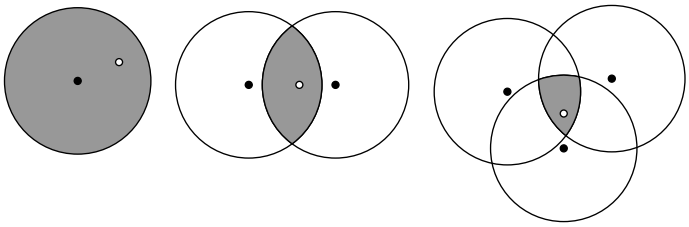


Fig. 18 Combination of constraints (adapted from [54])

The convex optimization algorithm can obtain a good performance with tight constraints. Moreover, experimental results show that the solution can be obtained quickly for a sensor network of several hundred nodes. Although the localization accuracy can be improved by the increase of connectivity constraints, the communication cost may increase significantly at the same time. That is a problem to be settled.

2.2.3 DV-Hop Algorithm

DV-hop [55] is a hop-by-hop positioning algorithm based on ad hoc positioning systems (APS). It is a combination of distance vector routing and GPS positioning to provide estimated positions for all the nodes in network. APS is suitable for indoor localization applications in which the main feature of the network is ad hoc deployment rather than the unpredictable topology. The following part will explain the APS scheme.

Ad Hoc Positioning System (APS)

Assume that WSN nodes are vertexes in a graph. They are connected efficiently, and the lengths of edges are known. Thus, the topology of this graph is known. Those nodes that stay still may act as beacon nodes, constituting a reference system of the network. The other nodes can locate themselves by the reference system using the trilateration techniques.

It is possible for the device to estimate the distance to its neighbors via received signal strength. The APS algorithm employs hop-by-hop propagation to obtain the distance between a sensor node to a beacon node. The immediate neighbors of the beacon nodes estimate their distance to the beacon nodes by signal strength measurements. The second hop neighbors are also able to obtain their distance to the beacon nodes via its one-hop neighbors, and the rest nodes of the network follow as a flood manner in control. Once a node in the network knows its distance to more than three beacon nodes, its position can be obtained by the trilateration techniques. However, APS is vulnerable to signal strength measurements.

DV-Hop Scheme

APS applies the DV-hop method in hop-by-hop distance propagation. In this scheme, firstly all nodes obtain their distance to the the beacon nodes in hops via vector exchange. Each node is represented by a table (x_i, y_i, h_i), where h_i is the hop count of the node from a specific beacon node, and it communicates and updates its table merely with its neighbours. Secondly, every beacon node estimates the average distance of one hop, which is defined as a correction of the whole network, when it knows its distance to the other beacon node. Finally, the sensor node can estimate its distance to the beacon node in meters according to the correction and the hop information. The correction that obtained by a beacon node (x_i, y_i) is shown as below

$$c_i = \frac{\sum\limits_{j \neq i} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum\limits_{j \neq i} h_j} \quad j \neq i \tag{66}$$

where both i and j are beacon nodes. The sensor node in the whole network chooses the correction of the nearest beacon node as the average distance of one hop to estimate its position.

Figure 19 illustrates the DV-hop scheme. The goal is to estimate the position of sensor node A on a basis of beacon nodes L_1, L_2, L_3 of the network. As is seen from the figure, the distances between L_1, L_2, L_3 are known. Then, each correction of L_1, L_2, L_3 (estimated average distance of one hop in meters) can be calculated as:

$$c_1 = \frac{75 + 150}{2 + 5} = 32.1 \quad c_2 = \frac{75 + 60}{2 + 2} = 33.8 \quad c_3 = \frac{60 + 150}{2 + 5} = 30 \tag{67}$$

L_2 is the closest beacon node to A, and therefore c_2 is chosen as the correction for A to estimate its position. As a consequence, the distances from A to L_1, L_2, L_3 are 33.8×2, 33.8, 33.82×3, respectively. Finally, A's position can be obtained by the trilateration techniques just as GPS does.

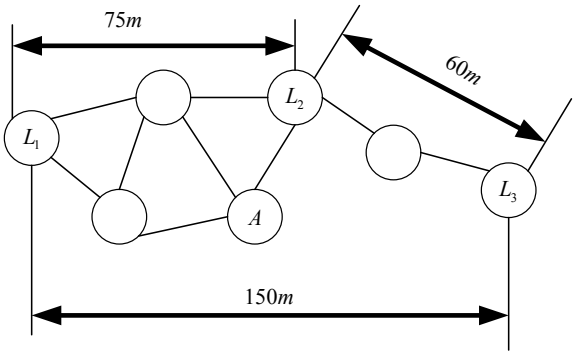


Fig. 19 The DV-hop scheme (adapted from [55])

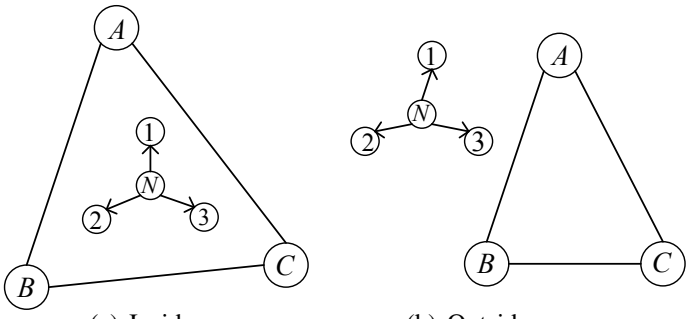


Fig. 20 The APIT test (adapted from [56])

The DV-hop algorithm is simple and hop-based. That is why it is invulnerable to measurement errors. Moreover, it does not require centralized computation that leads to a huge cost. The DV-hop scheme is fit to isotropic networks very well, but in irregular environments and networks with low density of beacon nodes, the localization accuracy may degrade significantly.

2.2.4 Approximate Point-in-Triangular Algorithm

The approximate point-in-triangular (APIT) algorithm [56] is a novel range-free scheme dividing the deployment area into many triangular regions to localize a sensor node, which estimates its possible position area by judging whether it is inside or outside of the triangles. The APIT algorithm mainly has two steps: (1) APIT test and (2) APIT aggregation. These two steps are implemented on an individual sensor node and will be described as follows.

APIT Test

In the APIT test, a sensor node first collects signals of beacon nodes. Then, three beacon nodes are randomly chosen from the collection to form a triangle. The basic idea of APIT test is that it tests the node's position via neighboring information. The inside and outside cases are shown in Fig. 20.

In Fig. 20a, it presents the case that none of the neighbor of node N (say, node 1, 2 and 3) reports close to or far away from all the three beacon nodes A, B, C at the same instant. In this case, sensor node N considers that it is inside the triangle. On the contrary, Fig. 20b shows another scenario that neighbor 2 claims to node N that it has longer distance to the three beacon nodes than N. As a consequence, N considers itself outside the triangle. The sensor node whose position is to be decided can collect signals from any beacon node within its communication range. Any three of these beacon nodes can form a triangle with each beacon node as the vertex.

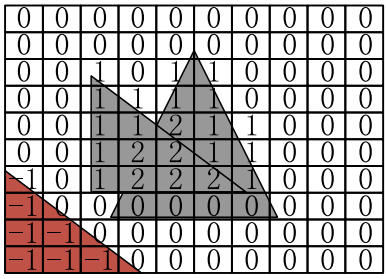


Fig. 21 The APIT aggregation approach (adapted from [56])

APIT Aggregation

Before the APIT aggregation, the APIT test is repeated until all the beacon nodes of the collection mentioned in section "APIT Test" are involved. Then, the results are aggregated by a grid scan approach that applies a grid to represent the area in which the sensor node is most likely to be. As shown in Fig. 21, for the inside decision of the APIT test, that is to say, the region inside an overlap of two triangles, the values of the region increase by one. Meanwhile, the values for the outside decision regions decrease. After all the triangles are taken into consideration, the grid regions with the maximum values (the grids with values of 2 in Fig. 21) are chosen as the final regions, and their center of gravity is decided as the the localization of the sensor node.

Different from the DV-hop algorithm, APIT can achieve high localization accuracy in anisotropic networks where irregular radio pattern exists. Also, the communication cost is low. However, it needs high density of beacon nodes due to the exchange of neighboring information. The localization accuracy of APIT can reach $0.5r$ under the conditions that $n_b = 16$, $n_d = 8$, where r is the communication range of the sensor node, n_b is the average number of beacon nodes in communication with the sensor node, and n_d is the number of sensor nodes on average per node radio area.

2.3 Range-Free Localization for Mobile Networks

The range-free algorithms mentioned in Sect. 2.2 work well for a WSN where all the nodes keep static after deployment. However, the performance may degrade largely along with the mobility of the nodes. Due to this problem, there are many localization algorithms for mobile networks that have been proposed. Monte Carlo localization (MCL) is one of the most popular algorithm among them.

2.3.1 Monte Carlo Localization

The Monte Carlo localization (MCL) [57] is based on probabilistic models of Markov chains. It was firstly used in localization for robots. In MCL methods, time is divided into discrete units. The position of a node at a specific moment is estimated based on the previous moment. During the estimation, the sequential Monte Carlo (SMC) method is utilized. It applies a set of weighted samples to represent the posterior distribution of a sensor's location and updates the samples recursively. The main steps of MCL algorithm are shown as follows:

- **Initialization**: Selecting N_{sample} samples from the deployment area as the initial set of the possible positions for the sensor node. N_{sample} is a constant that is preset.
- **Prediction**: The new possible position set L_t is obtained by the previous position l_{t-1} which is also estimated.

$$L_t = \{l_t^{\,i} | l_t^{\,i} \text{ is selected from } p(l_t | l_{t-1}, 1 \leq i \leq N_{sample})\}$$

- **Filtering**: Observations are obtained by the sensor nodes from their neighboring beacon nodes as filtering conditions. The samples satisfying the filtering condition get a weight value of 1, while other samples obtain a weight value of 0. Finally, the samples with the weighted value 0 are discarded by L_t.

$$L_{filtered} = \{l_t^{\,i} | l_t^{\,i} \text{ that } l_t^{\,i} \in l_t \text{ and } w_i > 0, w_i \text{ is the weighted value}\}$$

- **Resampling**: After the filtering, the sample number may be less than N_{sample}. Repeating the prediction and filtering steps until the sample number reaches N_{sample}.

Prediction

In the prediction step, the possible position samples at time unit t are collected based on the location estimated in the previous time unit, l_{t-1}. It is assumed that the maximum speed of the node is v_{max}, and its unit is meters per time interval. Then, the possible position area at time unit t is a circle with the origin l_{t-1} and radius v_{max}. If the speed of the node is uniformly distributed in $[0, v_{max}]$, the posterior probability of l_t is uniformly distributed in the circle, as (68) shows (Fig. 22):

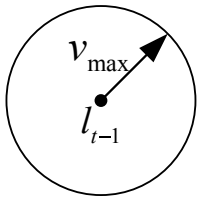


Fig. 22 Sampling of prediction

$$p(l_t|l_{t-1}) = \begin{cases} \dfrac{1}{\pi v_{\max}^2} & d(l_t, l_{t-1}) < v_{max} \\ 0 & d(l_t, l_{t-1}) \geq v_{max} \end{cases} \tag{68}$$

Filtering

In this step, each sample will get a weighted value 0 or 1 based on the filter condition. Then, the impossible position samples with weighted values 0 are filtered out. It is assumed that any node in communication with the beacon is in the radio range of the beacon node. For a sensor node, there are four types of beacon nodes to be considered.

- **outsiders**: beacon nodes whose signals were not received in both the previous and the current time unit.
- **arrivers**: beacon nodes whose signals were not received in the past time unit, but in the current.
- **leavers**: beacon nodes whose signals were received in the past time unit, but not in the current.
- **insiders**: beacon nodes whose signals were received in both the previous and the current time unit.

The filtering condition is mainly based on the information of arrivers and leavers because they provide the position changes of the sensor node. The arrivers within distance r (radio range) of a sensor node is called one-hop beacon nodes to the sensor node. Similarly, for the outsiders whose signals are received by the sensor node's neighbors (not the node) are called two-hop beacon nodes. The distance between the sensor node and two-hop beacon node is in the interval $(r, 2r)$. The filter condition of l is shown as follows:

$$w_i = \forall b \in S, d(l, b) \leq r \wedge \forall b \in T, r < d(l, b) \leq 2r \tag{69}$$

where b represents a beacon node, l is a sample of the location set L_t. S and T denote the one-hop and two-hop beacon neighbors of the sensor node, respectively. If the filter condition meets, the weighted value of the sample $w_i = 1$, otherwise, $w_i = 0$.

After iteratively repeating the prediction and filtering steps, the number of samples will reach N_{sample}. MCL algorithm achieves an estimation of the location of the sensor node at time t:

$$l_t = \frac{\sum_{i=1}^{N} l_t^i}{N_{sample}} \tag{70}$$

Simulation results show that the performance of MCL depends on the density of beacon nodes s_d (s_d is the number of beacon nodes on average within one hop communication range). When $s_d = 3$, $v_{max} = 0.2r$, and $N_{sample} = 50$, the localization accuracy can achieve $0.3r$. However, there are two shortcomings of the MCL algorithm. One is its low sampling efficiency, this may lead to more computation

cost. The other one is that it needs high beacon density to achieve high localization accuracy. Therefore, we also provide weighted MCL (WMCL) as an alternative algorithm.

2.3.2 Weighted MCL

The weighted MCL algorithm [58] is based on the MCL algorithm. It incorporates the same steps from MCL, but it improves the accuracy of localization when nodes move fast. In the prediction step, WMCL builds a bounding-box of the sampling area to improve the sampling efficiency. In the filtering step, WMCL introduces the information of one-hop sensor neighbors (not only beacon neighbors) to restrict the filter condition.

Constructing the Bounding-Box

The bounding-box is a sampling area that is restricted by a rectangle. Assuming that there are n one-hop beacon neighbors of a sensor node s, and the radio range of each sensor node is r. Then, the bounding box can be initiated as follows:

$$\begin{aligned} x_{min} &= max_{i=1}^{n}\{x_i - r\},\ x_{max} = min_{i=1}^{n}\{x_i + r\} \\ y_{min} &= max_{i=1}^{n}\{y_i - r\},\ y_{max} = min_{i=1}^{n}\{y_i + r\} \end{aligned} \tag{71}$$

After the initiation of the bounding-box, it can be further reduced according to the information of its one-hop neighbors. Firstly, as shown in Fig. 23, suppose that p is the two-hop beacon neighbor of s. The area in shadow can be removed from the sampling area. Otherwise, p will be the one-hop of s if the shadowed area is contained.

Furthermore, assuming that the maximum localization error of each sensor node in x-axis is ER_x, similarly, ER_y represents the maximum localization error in y-axis. p is assumed to be the one-hop sensor neighbor of s which is denoted as $p \in US(s)$, then we have:

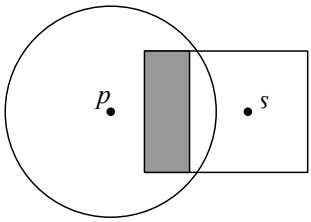


Fig. 23 Reduce of the bounding-box (adapted from [58])

$$
\begin{aligned}
|r_{t,p} - r_{t,s}| &\leq r \\
|r_{t,s} - r_{t-1,s}| &\leq v_{max} \\
|x(r_{t-1,s}) - x(e_{t-1,s})| &\leq ER_{x,t-1}(q) \\
|y(r_{t-1,s}) - y(e_{t-1,s})| &\leq ER_{y,t-1}(q)
\end{aligned} \tag{72}
$$

where $r_{t,s}$ and $e_{t,s}$ represent the real position and estimated position of sensor s in the time unit t, respectively, and $x(r_{t,s})$ and $y(r_{t,s})$ denote the x and y values of $r_{t,s}$ in the coordinate system, respectively. The bounding box built by (71) can be reduced further as (x'_{min} is taken as an example, notice that in (71) we have four parameters):

$$
x'_{min} = max\{x_{min}, max_{p \in US(s)}\{x(e_{t-1,p}) - v_{max} - r - ER_{x,t-1}(p)\}\} \tag{73}
$$

Weighting the Samples

In the filtering step in WMCL, each sample also obtains a weighted value as in MCL. The major difference is that WMCL also considers one-hop sensor node neighbors, not only beacon neighbors. As a result, WMCL can achieve high localization accuracy even with low density of beacon nodes. In WMCL, the weighted value of a sample is computed as:

$$
w_t^i = \prod_{s \in S} p(s|l_t^i) \prod_{s \in T} p(s|l_t^i) \prod_{s \in US} p(s|l_t^i) \tag{74}
$$

where S and T are the one-hop and two-hop beacon neighbor sets, US is the one-hop neighbor set. In (74), the first two parts are easy to compute just as MCL do. If $s \in S$, then we have:

$$
p(s|l_t^i) = [d(l_t^i, s) \leq r] \tag{75}
$$

If $s \in T$, we also have:

$$
p(s|l_t^i) = [r < d(l_t^i, s) \leq 2r] \tag{76}
$$

Nevertheless, the last part of the (74) is not easy to compute because the observation is not the real position of node s in this case. There is an approximate method to compute it. The method is shown in Fig. 24. It gives an approximation that the probability of $r_{t,s}$ in area B can be represented as the probability of $r_{t-1,s}$ in area A.

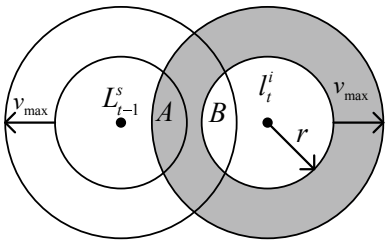


Fig. 24 The illustration of the approximate method (adapted from [58])

If $s \in US$, then we have:

$$p(s|l_t^i) = Pr\{l \in L_{t-1}^s \wedge l \in A\} \tag{77}$$

Furthermore, suppose that there are N samples in L_{t-1}^s, N' of the samples are in area A. The (77) can be simply computed as:

$$p(s|l_t^i) \approx \frac{N'}{N} \tag{78}$$

Different from other localization algorithms based on SMC, The WMCL algorithm works well both in static and mobile networks, especially when nodes move very fast. Moreover, it outperforms other algorithms with its high sampling efficiency and both low communication cost and computation cost. The sampling efficiency can be improved by 95% compared to MCL. When $s_d = 10$, $v_{max} = 0.2r$, and $N_{sample} = 50$, the localization error is about $0.2r$ less than that of MCL [57].

Notice that Sect. 2 mainly discusses localization approaches for cooperative sensor nodes within the WSN. Although the target is noncooperative in most cases, these approaches can still be applied (however modification is needed more or less) for target localization.

3 Target Tracking in Wireless Sensor Networks

Target tracking has always been an inherent purpose of WSN design. Both target detection (see Sect. 1) and node localization (see Sect. 2) provide a basis for target tracking to some extent. Recent years have seen the boom in various approaches of target tracking. In this section, typical tracking methods in WSN will be presented from two aspects. First, we summarize four categories of target tracking protocols, and then we propose three point track fusion approaches on a basis of prior target detection and localization approaches.

3.1 Target Tracking Protocols

This section investigates four major types of tracking protocols: (1) tree-based, (2) prediction-based, (3) hybrid methods, and (4) cluster-based. Of course, although this book only describe four categories due to the page limit, other types of protocols, for example, mobicast-message-based, are not included. Interested readers may refer to [59].

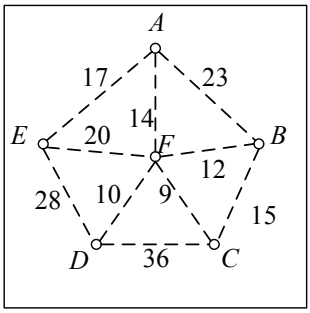


Fig. 25 The network graph (adapted from [62])

3.1.1 Tree-Based Tracking

In tree-based tracking, the topology of the network is represented as a graph, and all the nodes and their links are organized in a tree. The deviation avoidance tree (DAT) tracking protocol [62] is one of the examples. It develops a data aggregation model considering the physical topology of the sensor network and requires less total communication cost. There are two stages in the DAT protocol: DAT algorithm and query cost reduction (QCR). The first stage employs both deviation-avoidance principal and highest-weight-first principal to reduce the location update cost. The second stage tries to reduce the query cost by adjusting the tree built in the first stage.

DAT Algorithm

As illustrated in Fig. 25, the topology of a network is represented by a weighted graph $G = (V_G, E_G)$, where V_G stands for sensor nodes, and E_G is the set of edges between them. The weight on each edge is denoted by $w_G(u, v)$, $(u, v) \in E_G$. $w_G(u, v)$ is the sum of the target's event rates from u to v and v to u. The goal of DAT is to build an object tracking tree $T = (V_T, E_T)$ such that the location update cost is the lowest. In DAT algorithm, $V_T = V_G$, each sensor node in V_T is initiated as a singleton subtree and $E_T = \varnothing$. Then, E_G is rearranged in a decreasing order according to the weight, the new set is denoted by L_G. For each link (u, v) in L_G, it is chosen into E_T when the conditions meet:

$$\begin{aligned} &\text{Condition 1} = (u = root(u)) \wedge (dist_G(u, sink) = dist_G(v, sink) + 1) \\ &\text{Condition 2} = (v = root(v)) \wedge (dist_G(v, sink) = dist_G(u, sink) + 1) \end{aligned} \tag{79}$$

There is a restriction in (79) that $root(u) \neq root(v)$, where $root(x)$ is the root of the subtree containing x, and $dist_G(x, y)$ denotes the minimum hop count in x and y. Sink is the root node of the whole tree.

Query Cost Reduction

After the object tracking tree is built, the QCR method is used to achieve a low query cost. It examines all the nodes in T from bottom to top. For a nonleaf node v, its children nodes will be cut and connected to $p(v)$ ($p(v)$ is the parent of v). The

new tree is denoted by T', and the amount of cost reduction can be represented as $\Delta C = C(T) - C(T')$. If ΔC is positive, T will be replaced by T'. If not, T will stay unchanged. For a leaf node v, the goal is to make $p(v)$ closer to the sink. It can be achieved by change node $v's$ link from its current parent $p(v)$ to its grandparent $p(p(v))$. Similarly, T' denotes the new tree, T will turn into T' when the amount of cost function ΔC is positive. Otherwise, T stays the same.

The DAT tracking protocol is an efficient energy-saving method for object tracking. It shows how to build a logical tree for a sensor network to reduce the total communication cost in tracking objects. Simulation results show better performance comparing to other tracking protocols, such as the naive and drain-and-balance schemes. Interested readers may refer to [60, 61] for other tree-based target tracking protocols.

3.1.2 Prediction-Based Tracking

The prediction-based tracking methods predict the movement of an target with reduced energy consumption. Typical protocols are prediction-based energy saving (PES) [63] and dual prediction-based reporting (DPR) [64]. In this section, we will describe the former. PES provides the balance between energy consumption and data missing rate. It incorporates a prediction model, a wake-up mechanism, and a recovery mechanism.

Prediction Model

In PES, only one sensor called current node is needed to monitor the target (other nodes will sleep). The current node predicts the movement of the target and chooses some other sensors, namely target nodes, to help track the target collaboratively. Suppose that the speed and direction of a target is constant during a small period, the network needs X seconds to sense the target and report its location every T seconds. As a consequence, the current node will take $T - X$ seconds to predict the movement of the target, and the sensor node that the target finally reaches is known as the destination node. Since the previous positions of a target may reflect its future tendency of movement, the prediction model assigns a weight to each of the previous positions, and future positions of the target are predicted according to these previous positions and their weights.

Wake-Up Mechanism

If the prediction errors lead to a missing target, the PES will apply a wake-up mechanism illustrated in Fig. 26. The dashed arrow represents the predicted path of the target.

Figure 26 illustrates how a number of target nodes are woken up to conduct the target tracking simultaneously in order to avoid missing a target. There are three nodes of the wake-up mechanism:

1. The target node only contains the destination node.
2. The target nodes consist of both destination node and the nodes on the route from the current node to the destination node.

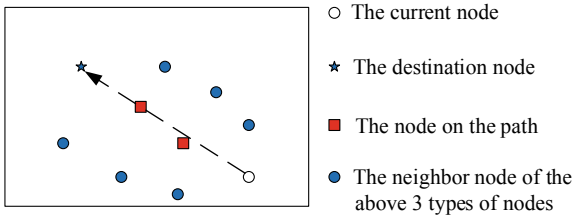


Fig. 26 The wake-up mechanism (adapted from [63])

3. The target nodes include the neighboring nodes surrounding the route, current node, and the destination.

The first and the second nodes consume less energy than that of the third; however, the third is more accurate in tracking performance.

Recovery Mechanism

The recovery mechanism is designed for relocating the target when it is missing. Firstly, the target nodes are woken up by the current node. If one sensor node finds the target, the current node will be notified and the recovery state finishes. If it fails, the flooding recovery begins so that all the nodes in the network are woken up to find the target. This ensures zero possibility of missing.

The PES protocol saves energy by minimizing the number of sensors involved, as well as the time of the active mode. Simulation results [63] show that PES can reduce the energy consumption of the microcontroller units efficiently. However, the assumed moving patterns are simple and not applicable in complex environments.

3.1.3 Hybrid Methods for Tracking

The hybrid method for tracking [65] assumes that the network is two-tier hierarchical. The nodes of the network are categorized into two levels: CHs and Normal Nodes (NNs). NNs can collect information of the environment and send it to their CHs without the capability of communicating with each other. CHs can send orders to NNs directly, and different CHs can communicate with each other.

The clusters have two states: *Active* and *Idle*. When the target appears or the future trajectory predicted by a neighbor CH intersects with its own coverage area, the cluster switches from *Idle* to *Active*. As for NNs, the additional states are *Probabilistic Active* and *Sleep*. The CH will assign *Probabilistic Active* or *Sleep* to NNs based on the estimated target location when the cluster is *Active*.

Let L_i be the location of node i and L_H stands for the positions of CH_i. Assume that the NNs are binary sensing models. r is the identical sensing range of NNs. If a target X appears at L_X, the sensing result R_i of i is:

$$R_i = \begin{cases} 1, ||L_i - L_X|| \leq r \\ 0, ||L_i - L_X|| > r \end{cases} \tag{80}$$

The centroid localization algorithm is used in the binary sensing model. Assume that the number of activated NNs is k, and the location of l ($l < k$) NNs which detect the target is $L_i(x_i, y_i)$, $i = 1, \ldots, l$. The estimated location is

$$L_e(t) = \left(\frac{\sum_{i=1}^{l} x_i}{l}, \frac{\sum_{i=1}^{l} y_i}{l} \right) \tag{81}$$

The rest of the NNs that fail to detect the target do not transmit the results to their CHs.

After localization, the movement prediction is carried out. It proposes a prediction algorithm by means of the recursive least square (RLS) technique [66]. At each sensing circle T_{sc}, the prediction procedures are as follows:

1. NN($t-1$) and NN(t) calculate local results and send them to their CH(t) (NN($t-1$) and NN(t) represent the previous and current active NNs respectively);
2. CH(t) fuses the results and calculates $L_e(t)$;
3. Predict the next location $L_p(t+1)$ by means of RLS;
4. Assign a new leader CH($t+1$) based on $L_p(t+1)$;
5. Deactivate NN($t-1$);
6. Activate NN($t+1$) with the probability of P;
7. Assign Subleaders according to the activated NNs.

The simulation shows that the larger the radius within which we activate the NNs near the predicted location, the lower the tracking error. The hybrid method can provide a nice balance between tracking accuracy and energy cost.

3.1.4 Cluster-Based Tracking

It has been proved in [43, 67] that node clustering in WSN offers advantages in both energy efficiency and lifetime extension. Sensors in a network can form clusters to perform tracking collaboratively. There have been a number of cluster-based tracking, such as DELTA [68], and distributed adaptive multi-sensor scheduling [69]. We shall describe them as follows.

DELTA Algorithm

The DELTA algorithm is a distributive approach via light measurements to dynamically build clusters for target tracking. The whole process of DELTA is shown in Fig. 27. There is a leader in a tracking cluster, i.e., CH. The function of the CH is to communicate with the base station, locate the target, and keep the coherence of the cluster, which is similar to the CH described in Sect. 1.3.2. Cluster members are the one-hop neighbors of CH. They need to report their tracking data to CH.

At first, all the sensor nodes are idle. When a target appears, the whole network turns into the election running mode immediately and a timer is set. Before the timer

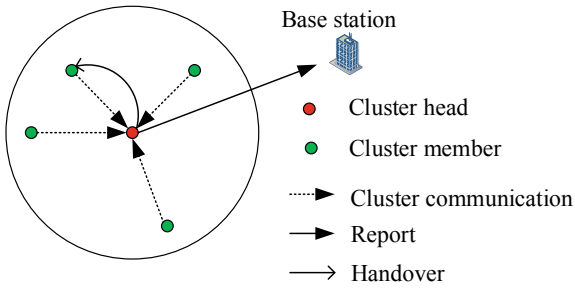


Fig. 27 Data flow of DELTA (adapted from [68])

ends, clusters and CHs are formed according to the light measurements. Then, the CH broadcast messages to inform their existence. At the same time, CHs mark state variables to ensure the update of a new cluster. Once the target is out of the sensing area of the cluster, the CH will broadcast a reelection message and turn to idle mode.

Different from other existing tracking protocols, the DELTA algorithm applies a passive heartbeat mechanism to overcome the restriction that all sensors' communication range should be higher than their sensing range. In the passive heartbeat mechanism, CH broadcasts heartbeat messages periodically to its cluster members. Then, the cluster members respond with the messages required by the CH. At the same time, these messages are heard by all the two-hop neighbors of CH so that they are aware of the existence of CH. When sensor nodes' sensing range is larger than communication range, the passive heartbeat mechanism avoids the forming of concurrent tracking clusters, otherwise it may incur confusion.

The DELTA algorithm is propitious to smart dust environments, in which the sensor has shorter radio range than sensing range. The CH election mechanism of DELTA is very fast and precise, providing an accurate tracking rate. However, the limitation is that the DELTA algorithm only works well in tracking a target with constant speed.

Distributed Adaptive Multi-sensor Scheduling

Distributed adaptive multi-sensor scheduling is an energy-saving algorithm based on the extended Kalman filter (EKF) and PES. The goal is to select sensors to perform target tracking task (tasking sensors) with reduced energy. Figure 28 describes a target tracking scenario in a WSN.

The distributed adaptive multi-sensor scheduling process can be described as follows.

1. Set the node deployment density to ensure a high probability of successful detection of a target. The target will be detected if the detection probability is larger or equal than the detection probability threshold.
2. Select the sampling interval. Calculate the optimal sampling interval via time update. The maximum sampling interval can be calculated from $\phi(k+1|k) = \phi_0$, where ϕ_0 is the given tracking accuracy and $\phi(k+1|k)$ is the predicted position uncertainty.

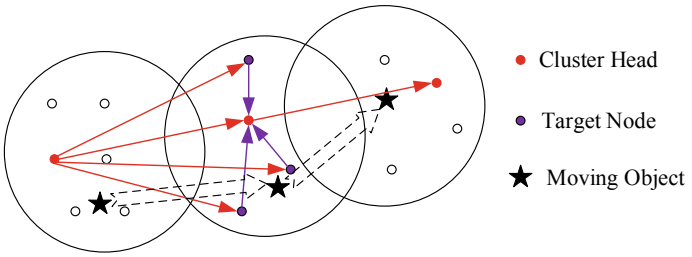


Fig. 28 Target tracking in the network (adapted from [69])

3. Select the tasking sensors. After selecting the next time steps sampling interval, tasking sensors will be also selected. the sensor with the maximum predicted detection probability (PDP) at the next time step is selected as the first member of the tasking cluster at time step $k + 1$.
4. Select the next CH. A CH energy measure (CHEM) for each selected tasking sensor is calculated by the current CH, and the tasking sensor with the lowest CHEM will be assigned next CH.
5. Apply Monte Carlo method to generate random samples of Gaussian distribution to simulate the random distribution of the predicted target state. PDP and joint detection probability (JDP) are estimated for selected tasking sensors.

Simulation results [69] show that this distributed adaptive multi-sensor scheduling saves the energy and achieves both tracking reliability and accuracy compared to single-sensor scheduling and multi-sensor scheduling with a uniform sampling interval.

3.2 Point Track Fusion

Apart from protocols in Sect. 3.1, there is an alternative methodology to investigate the tracking problem via WSN. Suppose sensor nodes are distributed, and each of them can independently localize a track point of the target. How to fuse all the points to obtain an accurate track? In this section, we propose three approaches: least norm (LN) algorithm, truncate Gaussian maximum likelihood (TGML) algorithm, and Gaussian maximum likelihood (GML) algorithm to answer this question.

Figure 29 illustrates the tracking model. Suppose two sensors i and j are measuring the location information of a target. These two sensors are at the location of (u_i, v_i) and (u_j, v_j), respectively. $T_m(x_m, y_m)$ denotes the real coordinate of mth target track point. Then $\{r_{i,m}, \theta_{i,m}\}$ and $\{r_{j,m}, \theta_{j,m}\}$ are the true values of ranges and angles between the target and sensors, respectively. $\{\Delta r_{i,m}, \Delta\theta r_{i,m}\}$ and $\{\Delta r_{j,m}, \Delta\theta r_{j,m}\}$ denote the measurement deviations of $\{r_{i,m}, \theta_{i,m}\}$ and $\{r_{j,m}, \theta_{j,m}\}$, respectively.

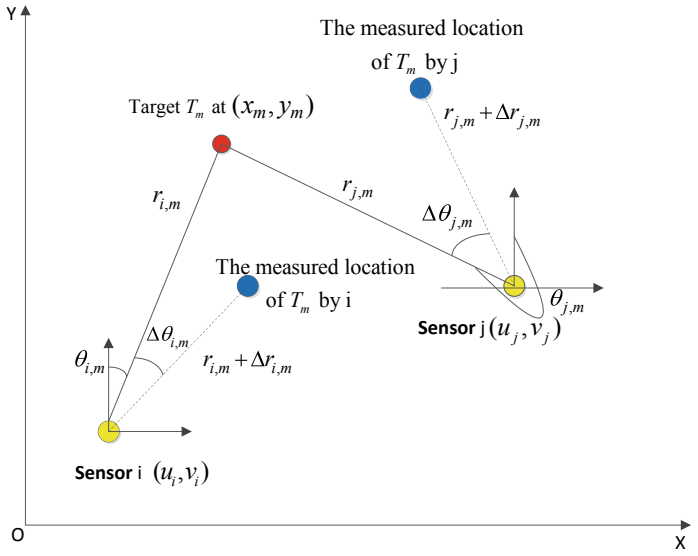


Fig. 29 Geometry model of point tracking fusion

3.2.1 Least Norm Algorithm

The LN algorithm assumes that the measurement deviations of different sensors are different. Moreover, the target tracking can be separated into a series of target track-point fusions. Based on approximations $\lim_{x\to 0}\cos(x) = 1$ and $\lim_{x_1,x_2\to 0} x_1 x_2 = 0$, the location formulas measured by different sensors can construct the following equations.

$$\begin{aligned}
&r'_{i,m}\cos(\theta'_{i,m})\Delta\theta_{i,m} + \sin(\theta'_{i,m})\Delta r_{i,m} - r'_{j,m}\cos(\theta'_{j,m})\Delta\theta_{j,m} - \sin(\theta'_{j,m})\Delta r_j \\
&= (r'_{i,m}\sin(\theta'_{i,m}) - u_i) - (r'_{j,m}\sin(\theta'_{j,m}) - u_j) \\
&-r'_{i,m}\sin(\theta'_{i,m})\Delta\theta_{i,m} + \cos(\theta'_{i,m})\Delta r_{i,m} + r'_{j,m}\sin(\theta'_{j,m})\Delta\theta_{j,m} - \cos(\theta'_{j,m})\Delta r_{j,m} \\
&= (r'_{i,m}\cos(\theta'_{i,m}) - v_i) - (r'_{j,m}\cos(\theta'_{j,m}) - v_j)
\end{aligned} \tag{82}$$

where $r'_{i,m}$, $\theta'_{i,m}$, $r'_{j,m}$, $\theta'_{j,m}$ denote the range and angle measured by sensors i and j $(i, j = 1, 2 \ldots K)$, respectively. Consequently, if we have K sensors, we will obtain $2(K-1)$ linear equations with $2K$ unknown variables, which can be represented by a matrix expression

$$\mathbf{L}_m \eta_m = \mathbf{x}_m \tag{83}$$

where $\eta_m = [\Delta\theta_{1,m}, \Delta r_{1,m}, \Delta\theta_{2,m}, \Delta r_{2,m}, \ldots, \Delta\theta_{K,m}, \Delta r_{K,m}]^T$,

$$\mathbf{x}_m = \begin{bmatrix} r'_{1,m}sin(\theta'_{1,m}) - r'_{2,m}sin(\theta'_{2,m}) + u_2 - u_1 \\ r'_{1,m}cos(\theta'_{1,m}) - r'_{2,m}cos(\theta'_{2,m}) + v_2 - v_1 \\ \vdots \\ r'_{1,m}sin(\theta'_{1,m}) - r'_{K,m}sin(\theta'_{K,m}) + u_K - u_1 \\ r'_{1,m}cos(\theta'_{1,m}) - r'_{K,m}cos(\theta'_{K,m}) + v_K - v_1 \end{bmatrix}. \tag{84}$$

Here we define the $\mathbf{L}_{S,m}$ and $\mathbf{L}_{i,m}(i = 1, 2, \ldots, K)$ as $\mathbf{L}_{S,m} = \begin{bmatrix} r'_{1,m}cos(\theta'_{1,m}) & sin(\theta'_{1,m}) \\ -r'_{1,m}sin(\theta'_{1,m}) & cos(\theta'_{1,m}) \end{bmatrix}$ $\mathbf{L}_{i,m} = \begin{bmatrix} -r'_{i,m}cos(\theta'_{i,m}) & -sin(\theta'_{i,m}) \\ r'_{i,m}sin(\theta'_{i,m}) & -cos(\theta'_{i,m}) \end{bmatrix}$. Then $\mathbf{L}_m$ can be expressed as

$$\mathbf{L}_m = \begin{bmatrix} \mathbf{L}_{S,m} & \mathbf{L}_{2,m} & 0 & 0 & \vdots & 0 \\ \mathbf{L}_{S,m} & 0 & \mathbf{L}_{3,m} & 0 & \vdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{L}_{S,m} & 0 & 0 & 0 & \cdots & \mathbf{L}_{K,m} \end{bmatrix} \tag{85}$$

Notice that η_m can be solved via the minimum norm because LN assumes the deviation is small. Thus, after obtaining the deviation of each sensor, the measurement error can be compensated. The mean of the estimation of K sensors is represented as the track of the target.

$$\eta_m = \mathbf{L}_m^T(\mathbf{L}_m\mathbf{L}_m^T)^{-1}\mathbf{x}_m \tag{86}$$

3.2.2 Truncate Gaussian Maximum Likelihood Algorithm

TGML algorithm assumes the system deviations follow a truncated Gaussian distribution, and the range $\Delta r_{k,m}$ and the angle $\Delta\theta_{k,m}$ are independent. The truncated Gaussian distribution means that the variable has a normal distribution within a truncated interval. Suppose the deviation obeys $\Delta r_{k,m} \sim N(0, \sigma_k^2)$ in $[-r_{k,m}, r_{k,m}]$, and $\Delta\theta_{k,m} \sim N(0, \sigma_k^2)$ in $[-\theta_{k,m}, \theta_{k,m}]$, the joint probability density function can be expressed as:

$$\begin{aligned} f(\Delta r_{k,m}, \Delta\theta_{k,m}) &= f(\Delta r_{k,m})f(\Delta\theta_{k,m}) \\ &= \begin{cases} c_{k,r}c_{k,\theta}\frac{1}{2\pi{\sigma_k}^2}e^{-\frac{\Delta r_{k,m}^2+\Delta\theta_{k,m}^2}{{\sigma_k}^2}}, & |\Delta r_{k,m}| \le r_{k,m}, |\Delta\theta_{k,m}| \le \theta_{k,m} \\ 0, & otherwise \end{cases} \end{aligned} \tag{87}$$

where $c_{k,r} = \frac{1}{2\varphi(\frac{r_{k,m}}{\sigma_k})-1}$, $c_{k,\theta} = \frac{1}{2\varphi(\frac{\theta_{k,m}}{\sigma_k})-1}$.

Suppose the kth sensor measures the coordinate of the mth target track point and obtains $(x_{k,m}, y_{k,m})$. $\Delta r_{k,m}$ and $\Delta\theta r_{k,m}$ can be expressed as

Table 1 Parameters selection of $\alpha_{k,m}$

$Sign(y_{k,m} - v_k)$	$Sign(x_{k,m} - u_k)$	
	+	−
+	0	2
−	1	1

$$\begin{aligned}\Delta r_{k,m} &= s_1(x_{k,m}, y_{k,m}) = \sqrt{(x_{k,m} - u_k)^2 + (y_{k,m} - v_k)^2} - \sqrt{(x_m - u_k)^2 + (y_m - v_k)^2} \\ \Delta\theta_{k,m} &= s_2(x_{k,m}, y_{k,m}) = (\text{atan}(\tfrac{x_{k,m}-u_k}{y_{k,m}-v_k}) + \alpha_{k,m}\pi) - (\text{atan}(\tfrac{x_m-u_k}{y_m-v_k}) + \beta_{k,m}\pi)\end{aligned} \tag{88}$$

where the values of $\alpha_{k,m}$ and $\beta_{k,m}$ are, respectively, listed in Tables 1 and 2.

Applying the method in [71], the joint probability density function can be obtained as

$$f(x_{k,m}, y_{k,m}) = f(\Delta r_{k,m}, \Delta\theta_{k,m})|J| = f(s_1, s_2)|J| \tag{89}$$

where $J = \det\begin{bmatrix} \frac{x_{k,m}-u_k}{\sqrt{(x_{k,m}-u_k)^2+(y_{k,m}-v_k)^2}} & \frac{y_{k,m}-v_k}{\sqrt{(x_{k,m}-u_k)^2+(y_{k,m}-v_k)^2}} \\ \frac{y_{k,m}-v_k}{(x_{k,m}-u_k)^2+(y_{k,m}-v_k)^2} & -\frac{x_{k,m}-u_k}{(x_{k,m}-u_k)^2+(y_{k,m}-v_k)^2} \end{bmatrix}$.

The objective optimization function which employs the maximum likelihood estimation is deduced in (90).

$$\max_{x_m, y_m} -\sum_{k=1}^{K} \frac{1}{2{\sigma_k}^2}(s_1^2 + s_2^2) \tag{90}$$

Though optimum solutions of parameters x_m and y_m cannot be solved directly, the rising gradient method in (91) is applied to find the approximate ones.

$$\begin{aligned} x_m^{(i+1)} &= x_m^{(i)} + \lambda g_x \\ y_m^{(i+1)} &= y_m^{(i)} + \lambda g_y \\ g_x &= \sum_{k=1}^{K} \tfrac{1}{{\sigma_k}^2}(s_1 \tfrac{x_m-u_k}{\sqrt{(x_m-u_k)^2+(y_m-v_k)^2}} + s_2 \tfrac{y_m-v_k}{\sqrt{(x_m-u_k)^2+(y_m-v_k)^2}}) \\ g_y &= \sum_{k=1}^{K} \tfrac{1}{{\sigma_k}^2}(s_1 \tfrac{y_m-v_k}{(x_m-u_k)^2+(y_m-v_k)^2} - s_2 \tfrac{x_m-u_k}{(x_m-u_k)^2+(y_m-v_k)^2}) \end{aligned} \tag{91}$$

Table 2 Parameters selection of $\beta_{k,m}$

$Sign(y_m - v_k)$	$Sign(x_m - u_k)$	
	+	−
+	0	2
−	1	1

3.2.3 Gaussian Maximum Likelihood Algorithm

In the GML algorithm, suppose both $\Delta r_{k,m}$ and $\Delta\theta_{k,m}$ obey the independent Gaussian distribution. Their joint probability density function is

$$f(\Delta r_{k,m}, \Delta\theta_{k,m}) = f(\Delta r_{k,m}) f(\Delta\theta_{k,m}) = \frac{1}{2\pi {\sigma_k}^2} e^{-\frac{\Delta r_{k,m}^2 + \Delta\theta_{k,m}^2}{{\sigma_k}^2}} \tag{92}$$

The GML algorithm can be considered as the special situation of TGML with truncated normal distribution in $[-\infty, \infty]$. Applying similar derivations in TGML, we find that GML cannot obtain the optimum solution with the likelihood estimation either. Similarly, applying the rising gradient method to find the approximate solutions we obtain the same expressions as (91).

3.2.4 Performance Evaluation of Point Track Fusion

In order to evaluate the target tracking performances of LN, TGML, and GML, firstly, the root mean square error (RMSE) between the fused estimated track and the real track is defined.

$$RMSE_X = \sqrt{\frac{\sum_{n=1}^{N} (\hat{x} - x)^2}{N}} \quad RMSE_Y = \sqrt{\frac{\sum_{n=1}^{N} (\hat{y} - y)^2}{N}} \tag{93}$$

In simulations, suppose all deviations of angles and ranges have the same normal distribution, i.e., $\Delta r_{k,m}, \Delta\theta_{k,m} \sim N(0, \sigma^2)$. Because $RMSE_Y$ has similar performances to $RMSE_X$, here we merely compare the performance of $RMSE_X$ for various sensor numbers in Fig. 30.

Figure 30 implies that increasing the number of available sensors gradually improves the RMSEs of LN, TGML, and GML. However, the RMSE of LN expresses a lower decline compared to those of TGML and GML when the sensors increase. This is because LN is deduced based on the first-order approximation with little derivations.

Figures 31 and 32 show the $RMSE_X$ over σ^2 when the sensor number is 5. Figure 31 illustrates that the RMSE of TGML is robust to truncated interval, and it is almost unchanged along with the increase of the truncated interval. On the other hand, the RMSE of LN tends to increase. Figure 32 shows that the larger the variance of the deviation, the larger the RMSE in GML. However, the RMSE of LN fluctuates along with the variance of the deviation.

Generally, simulations show that LN is robust in variances of the measurement deviation by sensors, and TGML is robust in the change of punctuated interval. For more details, interested readers can refer to [70].

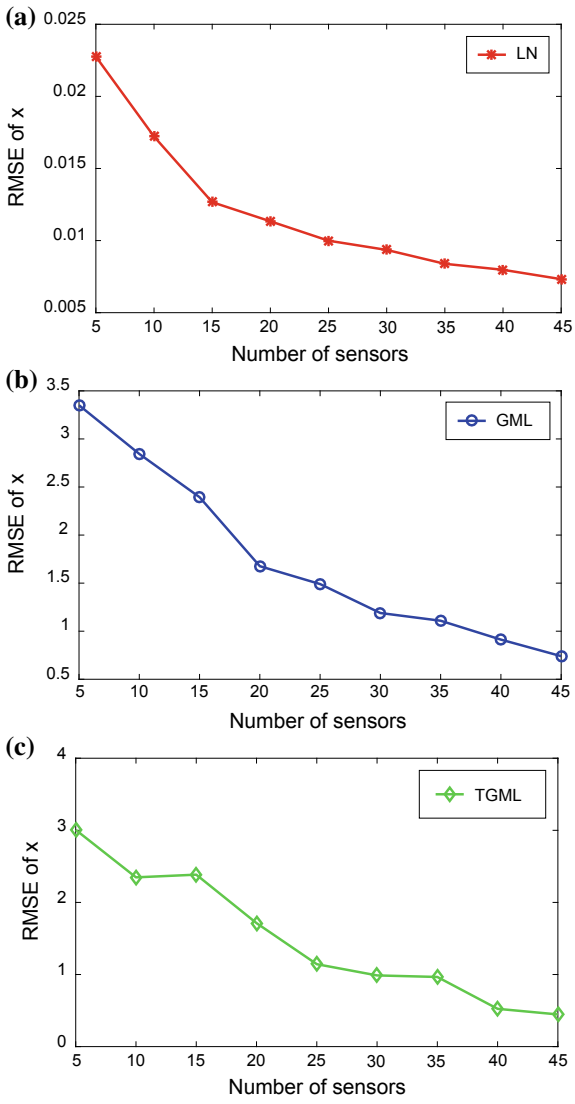


Fig. 30 The relationship curves of three algorithm between sensor number and RMSE. **a** LN algorithm in the condition of the deviation's variance $\sigma^2 = 10$, truncated interval $[-0.06, 0.06]$. **b** GML algorithm in the condition of $\sigma^2 = 10$. **c** TGML algorithm in the condition of $\sigma^2 = 10$, truncated interval $[-0.06, 0.06]$

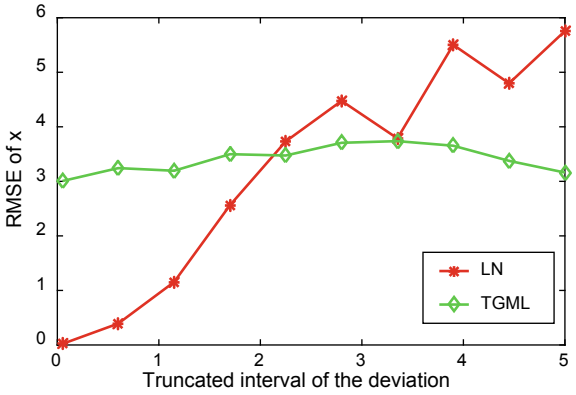


Fig. 31 TGML versus LN algorithm when WSN contains five sensors

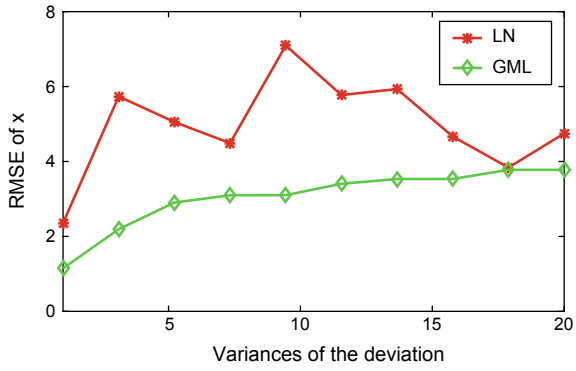


Fig. 32 GML versus LN algorithm when WSN contains five sensors

4 Conclusion

This chapter presents a comprehensive view of methods for target detection, sensor node localization, and target tracking in WSNs. Among the three applications, target detection and sensor node localization are the prerequisites for target tracking to some extend. We summarize the three parts as follows:

1. In the target detection part, coherent and noncoherent detection systems for distributed-RSNs are firstly introduced. Then, we compare the detection performance of two typical RSNs: distributed-RSN and MIMO-RSN. Simulation results show that distributed-RSN is superior to MIMO-RSN in Pd at both the same SNR and Pfa. To achieve the expected detection performance with low energy consumption, two nodes deployment strategies and two fuzzy clustering schemes are discussed. Under the same channel SNR, the DDS achieves higher probability of detection than HDS. Considering the two fuzzy clustering schemes, the F3&FCMSVDQR outperforms the F3&GORS in detection probability and the network's lifetime.

2. As for the sensor node localization, we categorize it into range-based and range-free approaches. The range-based schemes have higher localization accuracy; however, the cost including the communication and the computation cost can be huge. Thus range-free algorithms are more and more popular in recent years. The centroid, convex optimization localization, DV-hop, and APIT approaches are only suitable for localization in static WSNs. The localization of them may degrade significantly once the nodes in the network move. MCL and WMCL are promising techniques in applications of moving nodes. However, they do not perform well in harsh environments with some obstacles.
3. Four typical tracking protocols are discussed in the target tracking part. In the tree-based tracking, the topology of the network is represented as a graph, and all the nodes and their links are organized in a tree. Cluster-based approaches track the target by dynamically building clusters according to the movements of the target. Prediction-based algorithms depend on two schemes in addition that it needs to predict the target's movements. The hybrid can make a balance between tracking accuracy and energy cost. Besides, three point track fusion methods are introduced for the distributed WSNs when each node can get a track point of the target. Simulations show that LN is robust in variances of the measurement deviation by sensors, TGML is robust in the change of punctuated interval. The RMSE in the GML scheme increases with the variance of deviation.

Although we have given insights on the methods of target detection, sensor node localization, and target tracking, there still exist some problems for future research. Future work should be concentrated on application-oriented algorithms in these fields. Moreover, the real-time property and harsh environments should be taken into consideration.

References

1. Liang, J., Liang, Q.: Design and analysis of distributed radar sensor networks. IEEE Trans. Parallel Distrib. Syst. **22**(11) (2011)
2. Liang, J., Liang, Q., Zhou, Z.: Radar sensor network design and optimization for blind speed alleviation. In: Wireless Communications and Networking Conference 2007, pp. 2643–2647. 11–15 March 2007
3. Liang, Q.: Waveform design and diversity in radar sensor networks: theoretical analysis and application to automatic target recognition. IEEE Commun. Soc. Sensor Ad Hoc. Commun. Netw. **2**, 684–689 (2006)
4. Hung, D.L., Liang, Q.: Diversity in radar sensor networks: theoretical analysis and application to target detection. Int. J. Wirel. Inform. Netw. **16**(4), 209–216 (2009)
5. Dutta, P.K., Arora, A.K., Bibyk, S.B.: Towards radar-enabled sensor networks. In: Proceedings of the 5th International Conference on Information Processing in Sensor Networks, pp. 467–474 (2006)
6. Ly, H.D., Liang, Q.: Spatial-temporal-frequency diversity in radar sensor networks. In: Military Communications Conference (MILCOM 2006), Washington, DC, Oct 2006
7. Ly, H.D., Liang, Q.: Collaborative multi-target detection in radar sensor networks. In: IEEE Military Communications Conference, MILCOM 2007, pp. 1–7, 29–30 Oct 2007

8. Deng, H.: Target detection with distributed radar sensor networking systems (DRASENS). In: 2010 IEEE 10th International Conference on Signal Processing, pp. 1951–1954, 24–28 Oct 2010
9. Skolnik, M.I.: Introduction to Radar Systems, 3rd edn. McGraw Hill, New York (2001)
10. Levanon, N.: Radar Principles. Wiley, New York (1988)
11. Richards, M.A.: Fundamentals of Radar Signal Processing. McGraw Hill Companies, New York (2005)
12. Chevalier, F.L.: Principles of Radar and Sonar Signal Processing. Artech house, MA (2002)
13. Hume, A.L., Baker, C.J.: Netted radar sensing. In: Proceedings of the 2001 IEEE Radar Conference, pp. 23–26, May 2001
14. Haykin, S.: Cognitive radar networks. In: 2005 1st IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, pp. 1–3 Dec 2005
15. Dutta, P.K., Arora, A.K., Bibyk, S.B.: Towards radar-enabled sensor networks. In: The 5th International Conference on Information Processing in Sensor Networks, pp. 467–474, April 2006
16. Withington, P., Fluhler, H., Nag, S.: Enhancing homeland security with advanced radar sensors. IEEE Microwav. Mag. 51–58 (2003)
17. Fishler, E., Haimovich, A., Blum, R.S., Cimini, L.J., Chizhik, D., Valenzuela, A.: MIMO radar: an idea whose time has come. In: Proceedings of the IEEE Radar Conference, pp. 71–78, April 2004
18. Li, J., Stoica, P.: MIMO Radar Signal Processing. Wiley 2009
19. Xiao, L., Shi, J., Xiqi, G., Kai-kit, W.: Near-optimal power allocation for MIMO channels with mean or covariance feedback. IEEE Trans. Commun. **58**(1), 289–300 (2010)
20. Heliot, F., Imran, M.A., Tafazolli, R.: Energy-efficient power allocation for point-to-point MIMO systems over the Rayleigh fading channel. IEEE Wirel. Commun. Lett. **1**(4), 304–307 (2012)
21. Yellapantula, R., Yingwei, Y., Ansari, R.: Antenna selection and power control in MIMO systems with continuously varying channels. IEEE Commun. Lett. **13**(7), 480–482 (2009)
22. Liu, Y., Liang, J.: Distributed radar sensor network (RSN) versus MIMO-RSN. In: IEEE ICC 2013 RSN Workshop, pp. 911–915. Budapest, June 2013
23. Yang, L., Jing, L.: Optimization for distributed radar sensor network (RSN) and MIMO-RSN in flat fading channels. Phys. Commun. **13**, 253–259 (2014)
24. Mao, C., Liu, M., Liang, J., Zhao, G.: Performance for MIMO-RSN with different power allocation methods, pp. 2540–2544. IEEE ICCW, London (2015)
25. Ling, Y., Liang, J., Liu, W.: Graphical deployment Strategies in radar sensor networks (RSN) for target detection. EURASIP J. Wirel. Commun. Netw. **2013**, 55 (2013)
26. Liang, J., Hu, Y., Liu, H., Mao, C.: Fuzzy clustering in radar sensor networks for target detection. Ad Hoc Netw. (2016)
27. Shu, H., Liang, Q.: Data fusion in a multi-target radar sensor network. In: 2007 IEEE Radio and Wireless Symposium, pp. 129–132. 9–11 Jan 2007
28. Yu, S., Wang, R., Xu, H., Wan, W., Gao, Y., Jin, Y.: WSN nodes deployment based on artificial fish school algorithm for traffic monitoring system. In: IET International Conference on Smart and Sustainable City, pp. 1–5. 6-8 July 2011
29. Xu, K., Wang, Q., Hassanein, H., Takahara, G.: Optimal wireless sensor networks (WSNs) deployment: minimum cost with lifetime constraint. Wirel. Mobile Comp. Netw. Commun. **3**, 454–461 (2005)
30. Luo, H., Liu, Z., Xue, F.: A deployment strategy for target surveillance sensor networks based on acoustic energy measurements. In: 2nd International Conference on Future Computer and Communication, vol. 1, pp. 686–690. May 2010
31. Mageid, S.A., Ramadan, R.A.: Efficient deployment algorithms for mobile sensor networks. In: International Conference on Autonomous and Intelligent Systems, pp. 1–6, 21–23 June 2010
32. Lin, Y., Chen, B., Varshney, P.K.: Decision fusion rules in multi-hop wireless sensor networks. IEEE Trans. Aerosp. Electron. Syst. **41**(2) 2005

33. Aitsaadi, N., Achir, N., Boussetta, K., Pujolle, G.: Multi-objective WSN deployment: quality of monitoring, connectivity and lifetime. In: IEEE International Conference on Communications (ICC), pp. 1–6, 23–27 May 2010
34. Kapnadak, V., Coyle, E.J.: Optimal non-uniform deployment of sensors for detection in single-hop wireless sensor networks. In: 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), pp. 89–97. 27–30 June 2011
35. Gogu, A., Nace, D., Challal, Y.: A note on joint optimal transmission range assignment and sensor deployment for wireless sensor networks. In: 2010 14th International Telecommunications Network Strategy and Planning Symposium (NETWORKS), pp. 1–6. 27–30 Sept 2010
36. Xu, K., Hassanein, H., Takahara, G., Wang, Q.: Relay node deployment strategies in heterogeneous wireless sensor networks. IEEE Trans. Mobile Comput. **9**, 145–159 (2011)
37. Ababnah, A., Natarajan, B.: Optimal sensor deployment for value-fusion based detection. IEEE Global Telecommunications Conference, GLOBECOM 2009, pp. 1–6. Nov–Dec 2009
38. Zhao, F., Guibas, L.: Wireless Sensor Networks: an information processing approach, Morgan Kaufmann (2004)
39. Younis, O., Fahmy, S.: Distributed clustering in ad hoc sensornetworks: a hybrid, energy-efficient approach. In: Proceeding of IEEE Conference on Computer Communications (INFOCOM), pp. 629–640. Hong-Kong, China, Mar 2004
40. Ramaswamy, L., Gedik, B., Liu, L.: A distributed approach to node clustering in decentralized peer-to-peer networks. IEEE Trans. Parallel Distrib. Syst. **16**(9), 814–829 (2005)
41. Liang, Q.: Clusterhead election for mobile ad hoc wireless network. In: IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC2003). Beijing, Sept 2003
42. Wang, Y., Chen, L., Mei, J.-P.: Incremental fuzzy clustering with multiple medoids for large data. IEEE Trans. Fuzzy Syst. **22**(6), 1557–1568 (2014)
43. Frey, H., Görgen, D.: Geographical cluster-based routing in sensingcovered networks. IEEE Trans. Parallel Distrib. Syst. **17**(9), 899–911 (2006)
44. Kim, J.-M., Park, S.-H., Han, Y.-J., Chung, T.-M.: Chef: cluster head election mechanism using fuzzy logic in wireless sensor networks. In: 2008 10th International Conference on Advanced communication technology, (ICACT 2008), vol. 1, pp. 654–659 (2008)
45. Nayak, P., Devulapalli, A.: A fuzzy logic-based clustering algorithm for WSN to extend the network lifetime. IEEE Sens. J. **16**(1), 137–144 (2016)
46. Hu, Y., Liang, J.: CFAR decision fusion approaches in the clustered radar sensor networks using LEACH and HEED. Int. J. Distrib, Sens. Netw. (2015)
47. James, C.B.: Pattern recognition with fuzzy objective function algorithms. Kluwer Academic Publishers (1981)
48. Jung, J.W., Weitnauer, M.A.: On using cooperative routing for lifetime optimization of multi-hop wireless sensor networks: analysis and guidelines. IEEE Trans. Commun. **61**(8), 3413–3423 (2013)
49. Wei, C., Zhi, C., Fan, P., Letaief, K.B.: Asor: an energy efficient multi-hop opportunistic routing protocol for wireless sensor networks over rayleigh fading channels. IEEE Trans. Wirel. Commun. **8**(5), 2452–2463 (2009)
50. Hofmann-Wellenhof, B., Lichtenegger, H., Collins, J.: Global positioning system: theory and practice. Springer Science and Business Media (2012)
51. Akyildiz, I.F., Vuran, M.C.: Wireless Sensor Networks, 4th edn, Wiley (2010)
52. Niculescu D., Nath B.: Ad hoc positioning system (APS) using AOA. In: 2003 INFOCOM Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, vol. 3, pp. 1734–1743 (2003)
53. Bulusu, N., Heidemann, J., Estrin, D.: GPS-less low-cost outdoor localization for very small devices. IEEE pers. Commun. **7**(5), 28–34 (2000)
54. Doherty, L., El Ghaoui, L.: Convex position estimation in wireless sensor networks In: Proceedings—IEEE INFOCOM, vol. 3, pp. 1655–1663 (2001)

55. Niculescu, D., Nath, B.: DV based positioning in ad hoc networks. Telecommun. Syst. **22**(1–4), 267–280 (2003)
56. He, T., Huang, C., Blum, B.M., Stankovic, J.A., Abdelzaher, T.: Range-free localization schemes for large scale sensor networks. In: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, pp. 81–95 (2003)
57. Hu, L., Evans, D.: Localization for mobile sensor networks. Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, pp. 45–57 (2004)
58. Zhang, S., Cao, J., Li-Jun, C., Chen, D.: Accurate and energy-efficient range-free localization for mobile sensor networks. IEEE Trans. Mobile Comput. **9**(6), 897–910 (2010)
59. Chen, Y.-S., Liao, Y.-J.: HVE-mobicast: a hierarchical-variant-egg based mobicast routing protocol for wireless sensornets. In: IEEE Wireless Communications and Networking Conference (WCNC2006), vol. 2, pp. 697–702. Las Vegas, NV, USA, April 2006
60. Tsai, H.-W., Chu, C.-P., Chen, T.-S.: Mobile object tracking in wireless sensor networks. Comput. Commun. **30**, 1811–1825 (2007)
61. Kung, H.T., Vlah, D.: Efficient location tracking using sensor networks. In: IEEE Wireless Communications and Networking Conference (WCNC 2003). New Orleans, Louisiana, USA, March 2003
62. Lin, C.-Y., Peng, W.-C., Tseng, Y.-C.: Efficient in-network moving object tracking in wireless sensor networks. IEEE Trans. Mobile Comput. **8**(5), 1044–1056 (2006)
63. Xu, Y., Winter, J., Lee, W.-C.: Prediction-based strategies for energy saving in object tracking sensor networks. In: Proceedings of IEEE International Conference on Mobile Data Management, pp. 346–357 (2004)
64. Xu, Y., Winter, J., Lee, W.-C.: Dual predictionbased reporting for object tracking sensor networks. In: The 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous04), pp. 154–163 (2004)
65. Wang, Z., Li, H., Shen, X., Sun, X., Wang, Z.: Tracking and predicting moving targets in hierarchical sensor networks. In: IEEE International Conference on Networking, Sensing and Control, ICNSC, pp. 1169–1173 (2008)
66. Haykin, S.S.: Adaptive Filter Theory. Pearson Education India (2008)
67. Wang, Y., Chen, L., Mei, J.: Incremental fuzzy clustering with multiple medoids for large data. IEEE Trans. Fuzzy Syst. **22**(6), 1557–1568 (2014)
68. Wälchli, M., Skoczylas, P., Meer, M., Braun, T.: Distributed event localization and tracking with wireless sensors. In: Wired/Wireless Internet Communications, pp. 247–258 (2007)
69. Lin, J., Xiao, W., Lewis, F.L., Xie, L.: Energy-efficient distributed adaptive multisensor scheduling for target tracking in wireless sensor networks. IEEE Trans. Instrum. Meas. **58**(6), 1886–1896 (2009)
70. Yangyang H.: Research on signal fusion in heterogenous sensor network. Master thesis, University of Electronic Science and Tehnology of China (2016)
71. Degroot, M.H., Schervish, M.J.: Probability and Statistics. Pearson Education India (2012)

Regularization-Based Location Fingerprinting

Duc A. Tran

Abstract Location fingerprinting is an approach to GPS-free localization. This approach requires a prior training set of fingerprints sampled at known locations, by comparing to which the locations of future fingerprints can be determined. For good accuracy, the training set should be large enough to appropriately cover the area. However, in practice, a quality training set is not easy to obtain and as such recent studies have resorted to utilizing fingerprints that are available without location information; these are called unlabeled fingerprints. This chapter presents several ways one can use regularization to learn from unlabeled fingerprints. Regularization is a mathematical framework to learn a function from data by enforcing regularizers to improve generalizability. The following scenarios are discussed: (1) how the training set can be enriched with unlabeled fingerprints, (2) how a trajectory of a moving device can be computed given its sequential fingerprints, labeled or unlabeled, collected during the trajectory, and (3) how a device can be tracked in an online manner as it moves using real-time fingerprints.

1 Introduction

Location information is valuable to a myriad of applications of wireless networks. In a surveillance sensor network, it is crucial to know the location of an incident caught by a sensor, such as fire in a building or oil spill in a coastal water. The demand is also high for mobile apps providing navigation and other location-based services in hospitals, shopping malls, airport terminals, and campus buildings, to name a few. GPS is the most effective way to get location information but does not work indoors. Even for outdoor environments where this service is available, it is not energy-efficient to have to turn it on continuously all the time.

D. A. Tran (✉)
Department of Computer Science, University of Massachusetts Boston, Boston, MA, USA
e-mail: duc.tran@umb.edu

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_10

Consequently, numerous efforts have been made toward GPS-free localization solutions. A popular approach is to leverage a model correlating received signal strength (RSS) with distance [1]; this is called "ranging." Given a number of reference points (RPs), e.g., Wi-Fi access points [2] or FM broadcasting towers [3], we can locate a device by estimating its distances to these RPs based on RSS ranging and then using multi-lateration to compute the device's location. RSS ranging, however, is highly sensitive to noise interference [1]. Furthermore, radio propagates differently in different directions due to obstacles such as walls, people, and furniture. Positioning based on LED lighting [4] has also been proposed with promising accuracy, but as visible light does not penetrate physical obstacles, this technique is suitable only for short-range applications.

Location fingerprinting is a viable range-free localization alternative. An early adopter of this approach is RADAR [2], perhaps the world's first Wi-Fi RSS-based indoor positioning system. This system relies on a radio map, a lookup table that maps locations inside the building under localization to the RSS fingerprints empirically observed at these locations, respectively. The reference points are the Wi-Fi access points in the building. To locate a user, the radio map is searched to find the closest sample RSS reading and its corresponding location will be used as the estimate for the user's location. RADAR represents the fingerprint approach that uses kNN for comparison to the map [2, 5, 6]. One can also employ a model-based learning approach to relate a fingerprint to a location, for example, probabilistically using Bayesian inference [7] or non-probabilistically using Artificial Neural Networks [8] or Support Vector Machines [9–11].

To generalize, a fingerprint at a specific location is a vector of location-sensitive measurements observed about the mobile device at this location. For indoor environments, such a measurement can be a RSS reading from a nearby Wi-Fi access point [2], a FM broadcasting tower [3], or a cellular tower [12]. For underwater environments, a measurement can be a profile of echo-sounding signals transmitted from the device (e.g., an AUV) to the seafloor or ping signals to the surface buoys. In theory, any sensor information that is sensitive to location change, including sound [13], light [4], and geomagnetic field [14], can be included in the fingerprint vector. Combining different sensor data where applicable can lead to a rich set of discriminative features for the fingerprint information.

The fingerprint approach works on the basis that if fingerprint information is obtained for sufficiently many sample locations, then the device's location given a new fingerprint can be computed by comparing to these samples. Specifically, there are two phases: the training phase, which is often done offline, and the positioning phase, which is done online. In the training phase, a number of sample locations are surveyed to build a map pairing each location to a fingerprint. In the positioning phase, when we need to compute a location in real time, the fingerprint of the device is compared against the fingerprint map to find the best location match.

Despite its simplicity, the fingerprint approach is limited by the quality of the training data. The training data should be sufficiently large to be well-representative of the environment, both spatially and temporally; see example in Fig. 1. For a large area, many locations need to be surveyed to ensure good spatial coverage and many

Fig. 1 Example of surveyed area: Wi-Fi RSSI fingerprint data were obtained at 208 sample locations (shown as dots) on the computer science department floor (68 m × 63 m) at UMass Boston. There are in total 138 Wi-Fi access points, and from those unreachable, the corresponding RSSI is set to −100 db. At each sample location, the corresponding fingerprint is the average of the RSSIs observed at this location. RSSI was measured by a person carrying an Android phone in no particular heading direction

fingerprint readings need to be measured at each sample location to ensure good temporal coverage (the signal characteristics of the environment are not time-invariant). Consequently, the calibration task can be tedious and labor-intensive, causing bottleneck to localization accuracy.

To circumvent this problem, one can apply semi-supervised learning [15] to augment the (small) training dataset of fingerprints with non-training fingerprints (those available but without known location) [16–20]. Here, training fingerprints serve as labeled data and non-training fingerprints as unlabeled. Unlabeled fingerprints are abundant because they can easily be obtained for a mobile device without manual location labeling. In practice, fingerprints should have similar values at similar locations and differ at different locations. This spatial property can be useful to regulate

the learning. In Sect. 3, we present two different ways how location fingerprinting can be cast into a semi-supervised learning framework using regularization.

Another problem discussed in this chapter is location tracking of a mobile device based on its sequentially obtained fingerprints. As the device is moving, more fingerprints, labeled or unlabeled, may be obtained on the spot and we should utilize them to better locate the device beyond mere reliance on the prebuilt/enriched training data. Since these fingerprints are that of a trajectory, a useful observation is that if two fingerprints are measured in proximate times then so should their corresponding locations. Section 4 presents the formulation of fingerprint-based tracking as a regularization problem incorporating this kind of temporal smoothness to compute a moving device's trajectory from its sequential fingerprint history.

Any computing framework is meaningful only if it can be implemented efficiently in a real system. For fingerprint-based localization and tracking in real time, since the fingerprint stream can go to infinity, it is impossible to store and process all the fingerprints observed in the past due to storage and computation limits. It is therefore desirable to have an algorithm that can compute the location in an online manner using a manageable amount of memory and compute resources. One idea is to store and compute based on only a small set of representative fingerprints instead of all the observed fingerprints. The rationale is as follows. Since the location matrix of a mobile device over a time window exhibits a low-rank structure, as substantiated in [21], we conjecture that the fingerprint matrix over time should also be sparse because of the tight correspondence between a fingerprint and its location. Consequently, the fingerprint stream can be approximated by a sparse set of representative fingerprints with minimal loss of information. Section 5 presents an online algorithm based on this idea.

There are already numerous research works on mobile localization and tracking, but they make additional assumptions such that those about special sensors built in the device (e.g., gyroscope, accelerometer, compass, light sensor) [22], those about mobility-specific constraints (e.g, speed, predefined map) [6], and those that are network-specific (e.g., vehicular [23] or wireless sensor networks [24]). In contrast, the regularization frameworks presented in this chapter are aimed at universal applicability in the sense that they are orthogonally applicable to any type of fingerprint space; i.e., applicable where fingerprint information can be of radio signals, acoustic, visible light, or geomagnetic, etc., and can contain any other information so long as it is location-sensitive.

The remainder of this chapter is organized as follows. Section 2 presents some background about fingerprint localization, its state-of-the-art formulation, and solution using a training dataset. Section 3 discusses how to enrich the training data with unlabeled fingerprint information. Section 4 is focused on the trajectory reconstruction problem. Section 5 is about how a trajectory can be computed in a sequential manner. Finally, the chapter is concluded in Sect. 6.

2 Preliminaries

Denote the fingerprint space by $\mathscr{X} \subset \mathbb{R}^m$, where m is the number of fingerprint features, each taking a real-valued number; e.g., RSSI from different Wi-Fi APs, readings from inertial measurement units (accelerometer, gyroscope, magnetometer), and/or any other location-discriminative feature that is obtainable for the device. A fingerprint is said to be "labeled" if its ground-truth location is known and "unlabeled" otherwise. For a fingerprint $\mathbf{x}$, which is a point in this m-dimensional space, let $h(\mathbf{x})$ be the function indicating whether $\mathbf{x}$ is labeled ($h(\mathbf{x}) = 1$) or not ($h(\mathbf{x}) = 0$). Denote

$$y(\mathbf{x}) = \begin{cases} \text{ground-truth location of } \mathbf{x}, \text{ if } \mathbf{x} \text{ is labeled} \\ 0, \text{ if } \mathbf{x} \text{ is unlabeled.} \end{cases}$$

For the sake of presentation, we assume that the location space is 1D; hence, $y(\mathbf{x}) \in \mathbb{R}$. The case for higher dimensions is a trivial extension (where each coordinate is computed separately). The unknown in our formulation is a function $f : \mathscr{X} \to \mathbb{R}$ that returns the location estimate for a given fingerprint. Ideally, $f(\mathbf{x})$ should equal its ground-truth location $y(\mathbf{x})$ for every labeled $\mathbf{x}$.

Given a training set of labeled fingerprints, $T = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l\}$, one can use supervised learning to learn the location estimate f. For example, supervised learning can be formulated as a Tikhonov regularization problem minimizing the following regularized empirical risk

$$f^* = \underset{f \in \mathscr{H}_K}{\operatorname{argmin}} \sum_{i=1}^{l} (f(\mathbf{x}_i) - y(\mathbf{x}_i))^2 + \lambda_K \|f\|_K^2. \tag{1}$$

Here, the solution space for the location estimator f is the reproducing kernel Hilbert space (RKHS) $\mathscr{H}_K$ associated with a kernel function

$$\begin{aligned} K : \mathscr{X} \times \mathscr{X} &\to \mathbb{R} \\ (\mathbf{x}, \mathbf{x}') &\mapsto \exp\left(-\frac{|\mathbf{x} - \mathbf{x}'|^2}{2\gamma^2}\right) \end{aligned} \tag{2}$$

for some constant γ. The first term, $\sum_{i=1}^{l} (f(\mathbf{x}_i) - y(\mathbf{x}_i))^2$, represents the deviation compared to the ground truth, using the squared loss as an example. The second term, $\lambda_K \|f\|_K^2$, is added to enforce some property; in our case, f is preferred to be smooth with respect to kernel K. The notation $\|.\|_K$ denotes the norm in $\mathscr{H}_K$. Coefficient $\lambda_K > 0$ represents the enforcement weight of the regularization.

In our context, the RKHS $\mathscr{H}_K$ is a Hilbert space of real-valued functions defined on the fingerprint space $\mathscr{X}$, with the following properties. First, for every $\mathbf{x} \in \mathscr{X}$,

the function $K_{\mathbf{x}} \equiv K(\mathbf{x}, .) \in \mathscr{H}_K$. Second, which is referred to as the reproducing property, for every $\mathbf{x} \in \mathscr{X}$, we have $f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle$. In general, any symmetric positive definite function can be used for the kernel K, not just the Gaussian function defined in Eq. 2.

Because $\mathscr{H}_K$ is a vector space whose dimension can be infinite, trying to directly solve the minimization problem (1) is not computationally feasible. Fortunately, thanks to the beautiful theorem below, called the Representer Theorem [25], we can convert this problem to a minimization problem in a finite-dimensional space, which can be solved easily.

Theorem 1 (Representer Theorem) *Suppose we are given a non-empty set $\mathscr{X}$, a positive definite kernel $K : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$, a set of training samples $\{(\mathbf{x}_1, y(\mathbf{x}_1)), (\mathbf{x}_2, y(\mathbf{x}_2)), \ldots, (\mathbf{x}_l, y(\mathbf{x}_l))\}$, a strictly increasing function $g : [0, \infty) \to \mathbb{R}$, an arbitrary cost function $c : (\mathscr{X} \times \mathbb{R}^2)^l \to \mathbb{R} \cup \{\infty\}$. Then any $f \in \mathscr{H}_K$ minimizing the regularized risk functional*

$$c\left((\mathbf{x}_1, y(\mathbf{x}_1), f(\mathbf{x}_1)), (\mathbf{x}_2, y(\mathbf{x}_2), f(\mathbf{x}_2)), \ldots, (\mathbf{x}_l, y(\mathbf{x}_l), f(\mathbf{x}_l))\right) + g\left(\|f\|_K\right) \quad (3)$$

must lie in the subspace spanned by $\{K_{\mathbf{x}_1}, K_{\mathbf{x}_2}, \ldots, K_{\mathbf{x}_l}\}$.

Proof Project f onto the subspace spanned by the vectors $\{K_{\mathbf{x}_1}, K_{\mathbf{x}_2}, \ldots, K_{\mathbf{x}_l}\}$ to obtain

$$f = \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} + v$$

where $v \in \mathscr{H}_K$ is the orthogonal component; i.e,. $\langle v, K_{\mathbf{x}_i} \rangle = 0 \ \forall i = 1, 2, \ldots, l$. Because of the reproducing property,

$$\begin{aligned} f(\mathbf{x}_j) &= \langle f, K_{\mathbf{x}_j} \rangle \\ &= \left\langle \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} + v, K_{\mathbf{x}_j} \right\rangle \\ &= \sum_{i=1}^{l} \alpha_i \langle K_{\mathbf{x}_i}, K_{\mathbf{x}_j} \rangle + \langle v, K_{\mathbf{x}_j} \rangle \\ &= \sum_{i=1}^{l} \alpha_i K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned}$$

which is independent from v. Thus, the cost function $c(.)$ does not depend on v.

Next, we have

$$\begin{aligned}\|f\|_K^2 &= \left\langle \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} + v, \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} + v \right\rangle \\ &= \left\langle \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i}, \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} \right\rangle + \langle v, v \rangle \\ &= \left\| \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} \right\|_K^2 + \|v\|_K^2\end{aligned}$$

and so, as g is strictly increasing,

$$g(\|f\|) \geq g\left(\sqrt{\| \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} \|_K^2} \right);$$

the equality holds when $v = 0$. It becomes obvious that for f to minimize the risk in (3) we must have $v = 0$. It follows that f must lie in the subspace spanned by $\{K_{\mathbf{x}_1}, K_{\mathbf{x}_2}, \ldots, K_{\mathbf{x}_l}\}$.

Applying this theorem to our problem, where

$$c((\mathbf{x}_1, y(\mathbf{x}_1), f(\mathbf{x}_1)), (\mathbf{x}_2, y(\mathbf{x}_2), f(\mathbf{x}_2)), \ldots, (\mathbf{x}_l, y(\mathbf{x}_l), f(\mathbf{x}_l))) = \sum_{i=1}^{l} (f(\mathbf{x}_i) - y(\mathbf{x}_i))^2$$

and $g(x) = \lambda_K x^2$ (a strictly increasing function on $[0, \infty)$), hence

$$g(\|f\|_K) = \lambda_K \|f\|_K^2,$$

we will look for a solution of the form

$$f = \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i}$$

or

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i K(\mathbf{x}_i, \mathbf{x}) \; \forall \mathbf{x} \in \mathscr{X}$$

where the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_l \in \mathbb{R}$ are the only unknown to be found. Let us denote the following matrices:

- The location estimator vector

$$\mathbf{f} = \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \ldots \\ f(\mathbf{x}_l) \end{bmatrix}$$

- The kernel coefficient vector

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \ldots \\ \alpha_l \end{bmatrix}$$

- The label vector

$$\mathbf{y} = \begin{bmatrix} y(\mathbf{x}_1) \\ y(\mathbf{x}_2) \\ \ldots \\ y(\mathbf{x}_l) \end{bmatrix}$$

- The identity matrix $\mathbf{I} = diag(\underbrace{1, 1, \ldots, 1}_{l})$
- The kernel matrix

$$\mathbf{K} = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \ldots & K(\mathbf{x}_1, \mathbf{x}_l) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \ldots & K(\mathbf{x}_2, \mathbf{x}_l) \\ \ldots & & & \\ K(\mathbf{x}_l, \mathbf{x}_1) & K(\mathbf{x}_l, \mathbf{x}_2) & \ldots & K(\mathbf{x}_l, \mathbf{x}_l) \end{bmatrix}$$

We have

$$\sum_{i=1}^{l} (f(\mathbf{x}_i) - y(\mathbf{x}_i))^2 = (\mathbf{f} - \mathbf{y})^\intercal (\mathbf{f} - \mathbf{y}) = (\mathbf{K}\boldsymbol{\alpha} - \mathbf{y})^\intercal (\mathbf{K}\boldsymbol{\alpha} - \mathbf{y})$$

and

$$\begin{aligned} \|f\|_K^2 &= \langle f, f \rangle \\ &= \left\langle \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i}, \sum_{i=1}^{l} \alpha_i K_{\mathbf{x}_i} \right\rangle \\ &= \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j \langle K_{\mathbf{x}_i}, K_{\mathbf{x}_j} \rangle \end{aligned}$$

$$= \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$$
$$= \boldsymbol{\alpha}^\intercal \mathbf{K} \boldsymbol{\alpha}$$

The minimization problem in (1) can be expressed in matrix form as

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \ (\mathbf{K}\boldsymbol{\alpha} - \mathbf{y})^\intercal (\mathbf{K}\boldsymbol{\alpha} - \mathbf{y}) + \lambda_K \boldsymbol{\alpha}^\intercal \mathbf{K} \boldsymbol{\alpha}.$$

Assuming $\mathbf{K}$ is invertible, using gradient descent to solve this minimization problem, we can easily obtain

$$\boldsymbol{\alpha}^* = \begin{bmatrix} \alpha_1^* \\ \alpha_2^* \\ \cdots \\ \alpha_l^* \end{bmatrix} = (\lambda_K \mathbf{I} + \mathbf{K})^{-1} \mathbf{y}.$$

In the positioning phase, given a new fingerprint $\mathbf{x}$, its predicted location will be

$$f^*(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i^* K(\mathbf{x}_i, \mathbf{x}).$$

The localization error of this prediction depends on the quality of the training set T.

3 Enrichment of Training Data

Suppose that, in addition to the original training set $T = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l\}$, whose ground-truth location is known, we have a supplemental set of unlabeled fingerprints, $U = \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \ldots, \mathbf{x}_{l+u}\}$, whose location is unknown. In practice, $u \gg l$. If we can somehow learn the locations of the unlabeled fingerprints, the extended fingerprint map $\{y(\mathbf{x})\}_{\mathbf{x} \in T \cup U}$ with $n = l + u$ samples, instead of the original map $\{y(\mathbf{x})\}_{\mathbf{x} \in T}$ with only l samples, should be used for training purposes. Since the extended map is richer, it is highly likely that the localization quality during the positioning phase will be better.

One way to learn the locations of these unlabeled fingerprints based on the labeled is by applying Bayesian inference [26]. This method, called the generative method, consists in two steps. First, we estimate a distribution for the probability $p(\mathbf{x}|y)$ of observing a fingerprint $\mathbf{x}$ at a given location y. Second, the location corresponding to a fingerprint $\mathbf{x}$ is determined based on the probability

$$p(y|\mathbf{x}) \propto p(\mathbf{x}|y) p(y)$$

where $p(y)$ is the probability of location y. The location distribution is assumed known.

The generative method relies on choosing the right model for the distribution $p(\mathbf{x}|y)$, which is not easy. Alternatively, one can use the regularization method to propagate the location labels for the unlabeled fingerprints based on their similarity with the labeled. This makes sense because of the spatial smoothness in the fingerprint space. The de facto regularization framework for semi-supervised learning is Manifold Regularization proposed by Belkin et al. [27]. Pan et al. [16, 17] apply this framework to fingerprint localization, in which a Laplacian regularization term is added to regulate the intrinsic manifold structure of the fingerprints; here, the manifold is a weighted graph of fingerprints in which the weight of an edge connecting two fingerprints represents their similarity. Total Variation Regularization, which is an alternative framework for semi-supervised learning [28], has been explored for location fingerprinting in the work of Tran and Truong [19]. We present below the formulations using Manifold Regularization and Total Variation Regularization.

3.1 Manifold Regularization

The Manifold Regularization method for semi-supervised learning extends the regularization formulation (1) by introducing a Laplacian regularization term to enforce the smoothness with respect to an intrinsic manifold. To apply this framework to location fingerprinting, as in [16, 17], we need to solve the following problem:

$$\min_{f \in \mathcal{H}_K} \sum_{i=1}^{l} (f(\mathbf{x}_i) - y(\mathbf{x}_i))^2 + \lambda_K \|f\|_K^2 + \lambda_{MR} \sum_{i,j=1}^{n} w(\mathbf{x}_i, \mathbf{x}_j)(f(\mathbf{x}_i) - f(\mathbf{x}_j))^2. \tag{4}$$

Here, the intrinsic manifold is a similarity graph W of $n = l + u$ vertices, each representing a fingerprint. W can be constructed as a kNN-graph or an ε-ball-graph. As a kNN-graph, each vertex is connected to its k nearest vertices, i.e., those at smallest distances (distance in the fingerprint space). As an ε-ball-graph, each vertex is connected to every vertex within a distance bounded by ε. The distance can be defined based on any metric such as Euclidean or Manhattan, and the value k or ε chosen must ensure that the graph is connected. Once the edges are formed, each edge $(\mathbf{x}_i, \mathbf{x}_j)$ is associated with a weight $w(\mathbf{x}_i, \mathbf{x}_j)$ to represent the similarity between the fingerprints $\mathbf{x}_i$ and $\mathbf{x}_j$. The similarity measure is most often based on a Gaussian Radial Basis Function; i.e., $w(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{|\mathbf{x}-\mathbf{x}'|^2}{2\sigma^2}\right)$ for some σ. The weight is set to zero for non-edges.

The additional regularization term in (4) is called the Laplacian regularizer because we can express

$$\sum_{i,j=1}^{n} w(\mathbf{x}_i, \mathbf{x}_j)(f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 = \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \cdots \\ f(\mathbf{x}_n) \end{bmatrix} \mathbf{L} \begin{bmatrix} f(\mathbf{x}_1) & f(\mathbf{x}_2) & \ldots & f(\mathbf{x}_n) \end{bmatrix}$$

where $\mathbf{L}$ is the Laplacian matrix of the similarity graph W,

$$\mathbf{L} = \left[l_{ij} = \begin{cases} -w(\mathbf{x}_i, \mathbf{x}_j) & \text{if } i \neq j \\ \sum_{k=1}^{n} w(\mathbf{x}_i, \mathbf{x}_k) & \text{otherwise} \end{cases} \right]_{n \times n}$$

Let us denote $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_n)]^\intercal$, $\mathbf{y} = [y(\mathbf{x}_1), y(\mathbf{x}_2), \ldots, y(\mathbf{x}_n)]^\intercal$, $\mathbf{H} = diag\,(h(\mathbf{x}_1), h(\mathbf{x}_2), \ldots, h(\mathbf{x}_n))$, the identity matrix $\mathbf{I} = diag(\underbrace{1, 1, \ldots, 1}_{n})$, and the kernel matrix $\mathbf{K} = \left[k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)\right]_{n \times n}$. Then, we can write

$$\sum_{i=1}^{l} (f(\mathbf{x}_i) - y(\mathbf{x}_i))^2 = (\mathbf{f} - \mathbf{y})^\intercal \mathbf{H} (\mathbf{f} - \mathbf{y}).$$

Because we just need to find the best values for $\{f(\mathbf{x}_{l+1}), f(\mathbf{x}_{l+2}), \ldots, f(\mathbf{x}_n)\}$, it suffices to solve the following minimization:

$$\min_{\mathbf{f}} \left\{ J(\mathbf{f}) = (\mathbf{f} - \mathbf{y})^\intercal \mathbf{H} (\mathbf{f} - \mathbf{y}) + \lambda_K \|\mathbf{f}\|_K^2 + \lambda_{MR} \mathbf{f}^\intercal \mathbf{L} \mathbf{f} \right\}. \tag{5}$$

Proposition 1 *The minimizer of risk $J(\mathbf{f})$ in (5) admits the following solution:*

$$\mathbf{f}^* = (\lambda_K \mathbf{I} + \mathbf{K}\mathbf{H} + \lambda_{MR} \mathbf{K}\mathbf{L})^{-1} \mathbf{K}\mathbf{H}\mathbf{y}. \tag{6}$$

Proof Because f belongs to the RKHS $\mathscr{H}_K$, according to the Representer Theorem, we look for a solution in the form $\mathbf{f} = \mathbf{K}\boldsymbol{\alpha}$ for some column vector $\boldsymbol{\alpha} \in \mathbb{R}^n$. Therefore, $\|\mathbf{f}\|_K^2 = \boldsymbol{\alpha}^\intercal \mathbf{K} \boldsymbol{\alpha}$ and the risk $J(\mathbf{f})$ in Eq. (5) becomes: (note that $\mathbf{K}$ is symmetric and so $\mathbf{K}^\intercal = \mathbf{K}$)

$$\begin{aligned} J(\mathbf{f}) &= (\mathbf{f} - \mathbf{y})^\intercal \mathbf{H} (\mathbf{f} - \mathbf{y}) + \lambda_K \boldsymbol{\alpha}^\intercal \mathbf{K} \boldsymbol{\alpha} + \lambda_{MR} \mathbf{f}^\intercal \mathbf{L} \mathbf{f} \\ &= \mathbf{f}^\intercal (\mathbf{H} + \lambda_{MR} \mathbf{L}) \mathbf{f} - 2\mathbf{y}^\intercal \mathbf{H} \mathbf{f} + \mathbf{y}^\intercal \mathbf{H} \mathbf{y} + \lambda_K \boldsymbol{\alpha}^\intercal \mathbf{K} \boldsymbol{\alpha} \\ &= \boldsymbol{\alpha}^\intercal \mathbf{K} (\mathbf{H} + \lambda_{MR} \mathbf{L}) \mathbf{K} \boldsymbol{\alpha} - 2\mathbf{y}^\intercal \mathbf{H} \mathbf{K} \boldsymbol{\alpha} + \mathbf{y}^\intercal \mathbf{H} \mathbf{y} + \lambda_K \boldsymbol{\alpha}^\intercal \mathbf{K} \boldsymbol{\alpha} \\ &= \boldsymbol{\alpha}^\intercal \underbrace{\mathbf{K} \left(\lambda_K \mathbf{I} + (\mathbf{H} + \lambda_{MR} \mathbf{L}) \mathbf{K}\right)}_{\mathbf{Q}} \boldsymbol{\alpha} - 2\mathbf{y}^\intercal \mathbf{H} \mathbf{K} \boldsymbol{\alpha} + \mathbf{y}^\intercal \mathbf{H} \mathbf{y} \\ &= \boldsymbol{\alpha}^\intercal \mathbf{Q} \boldsymbol{\alpha} - 2\mathbf{y}^\intercal \mathbf{H} \mathbf{K} \boldsymbol{\alpha} + \mathbf{y}^\intercal \mathbf{H} \mathbf{y}. \end{aligned}$$

To minimize $J(\mathbf{f})$, set its derivative with respect to $\boldsymbol{\alpha}$ to zero,

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = (\mathbf{Q} + \mathbf{Q}^{\intercal})\boldsymbol{\alpha} - 2\mathbf{KHy} = 0.$$

Since $\mathbf{K}$, $\mathbf{H}$, and $\mathbf{L}$ are symmetric, we have $\mathbf{Q}^{\intercal} = \mathbf{Q}$ and so $2\mathbf{Q}\boldsymbol{\alpha} - 2\mathbf{KHy} = 0$ or $(\lambda_K \mathbf{I} + \mathbf{K}(\mathbf{H} + \lambda_{MR}\mathbf{L}))\,\mathbf{K}\boldsymbol{\alpha} - \mathbf{KHy} = 0$. Because $\mathbf{f} = \mathbf{K}\boldsymbol{\alpha}$, we obtain

$$\mathbf{f} = (\lambda_K \mathbf{I} + \mathbf{KH} + \lambda_{MR}\mathbf{KL})^{-1}\,\mathbf{KHy},$$

assuming the matrix inverse is possible.

3.2 Total Variation Regularization

Total Variation (TV) Regularization is a widely used regularization framework for restoring images in the area of image processing [28]. TV permits sharper edges near the decision boundaries, whereas the Laplacian regularization penalizes too much gradients on edges. The former's effectiveness as an alternative framework for semi-supervised learning has been demonstrated, for example, by Bresson and Zhang [29]. TV can be used to enrich the training quality for location fingerprinting, as first reported in the work of Tran and Truong [19].

In the TV framework for semi-supervised learning, the minimization problem is

$$\min_{f} \sum_{i=1}^{l} (f(\mathbf{x}_i) - y(\mathbf{x}_i))^2 + \lambda_{TV} TV_W, \tag{7}$$

where TV_W is the TV of the function f on the similarity graph W described earlier. By definition, the TV of a continuous function f is

$$TV[f] = \int_{\mathscr{X}} \|\nabla f\|\, dx,$$

where $\mathscr{X}$ is the domain (continuous) of f, ∇f is its gradient, and dx is the area element of Ω of f. This concept can be extended for weighted graphs as follows. On graph W, the TV of a real-valued scalar function f defined on its vertices is the sum of the local TV at each and every vertex,

$$TV_W = \sum_{i=1}^{n} \|\nabla f(i)\|_{L^p(w)}.$$

The local TV at vertex $\mathbf{x}_i$ is the weighted L^p-norm of the gradient at this vertex. The gradient of function f at vertex $\mathbf{x}_i$ is

$$\nabla f(\mathbf{x}_i) = \begin{pmatrix} f(\mathbf{x}_1) - f(\mathbf{x}_i) \\ f(\mathbf{x}_2) - f(\mathbf{x}_i) \\ \cdots \\ f(\mathbf{x}_j) - f(\mathbf{x}_i) \\ \cdots \\ f(\mathbf{x}_n) - f(\mathbf{x}_i) \end{pmatrix}$$

and so,

$$\|\nabla f(\mathbf{x}_i)\|_{L^p(w)} = \left(\sum_{j=1}^{n} w(\mathbf{x}_i, \mathbf{x}_j) |f(\mathbf{x}_j) - f(\mathbf{x}_i)|^p \right)^{1/p}.$$

The graph TV above is a generalization of that defined in [29] and [30]. Note that the case $p = 1$ corresponds to the graph TV used by Bresson and Zhang [29],

$$TV_W = \sum_{i,j=1}^{n} w(\mathbf{x}_i, \mathbf{x}_j) |f(\mathbf{x}_j) - f(\mathbf{x}_i)|,$$

and the case $p = 2$ corresponds to the graph TV used by Elmoataz et al. [30],

$$TV_W = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{n} w(\mathbf{x}_i, \mathbf{x}_j)(f(\mathbf{x}_j) - f(\mathbf{x}_i))^2}.$$

The minimization in (7) becomes

$$\min_f \sum_{i=1}^{l} (f(\mathbf{x}_i) - y(\mathbf{x}_i))^2 + \lambda_{TV} \left(\sum_{j=1}^{n} w(\mathbf{x}_i, \mathbf{x}_j) |f(\mathbf{x}_j) - f(\mathbf{x}_i)|^p \right)^{1/p}. \quad (8)$$

This problem can be solved generally using the algorithm in [30]. Alternatively, there is a simpler algorithm [19] which fixes the locations for the labeled (setting $f(\mathbf{x}_i) = y(\mathbf{x}_i)$ for labeled $\mathbf{x}_i$) and iteratively adjusts the location estimates for the unlabeled as long as the value of the TV regularization term continues to decrease. This algorithm works as follows:

1. Initial step: for $i, j \in [1, n]$

$$f_i^{(0)} = \begin{cases} y(\mathbf{x}_i) & \text{if } i \le l \\ 0 & \text{otherwise} \end{cases}$$
$$\gamma_{ij}^{(0)} = w(\mathbf{x}_i, \mathbf{x}_j)$$

2. Iterative step $t = 0, 1, 2, \ldots$: for $i, j \in [1, n]$

$$f_i^{(t+1)} = \begin{cases} f_i^{(t)} & \text{if } i \le l \\ \frac{\sum_{j=1}^{n} \gamma_{ij}^{(t)} f_j^{(t)}}{\sum_{j=1}^{n} \gamma_{ij}^{(t)}} & \text{otherwise} \end{cases}$$

$$\gamma_{ij}^{(t+1)} = \frac{w(\mathbf{x}_i, \mathbf{x}_j)}{\|\nabla f^{(t)}(i)\|_{L^2(w)}} + \frac{w(\mathbf{x}_i, \mathbf{x}_j)}{\|\nabla f^{(t)}(j)\|_{L^2(w)}}$$

3. Stop when $\sum_{i=l+1}^{n} |f_i^{(t+1)} - f_i^{(t)}| < \tau$ (a predefined threshold). When stopped, the value of $f_i^{(t)}$ is used as the estimated location for each unlabeled fingerprint $\mathbf{x}_i$.

3.3 *Manifold Versus Total Variation Regularization*

We present below the results of an evaluation to assess the effectiveness of Manifold Regularization and Total Variation Regularization for enriching the training fingerprint set. This evaluation, published in [19], uses a Wi-Fi fingerprint dataset obtained from an indoor experiment, courtesy of [9] (University of Trento), containing 257 RSSI fingerprints at 257 sample locations in a WLAN with six Wi-Fi access points (see floor plan in Fig. 2). The sample locations are regular-grid points of the floor.

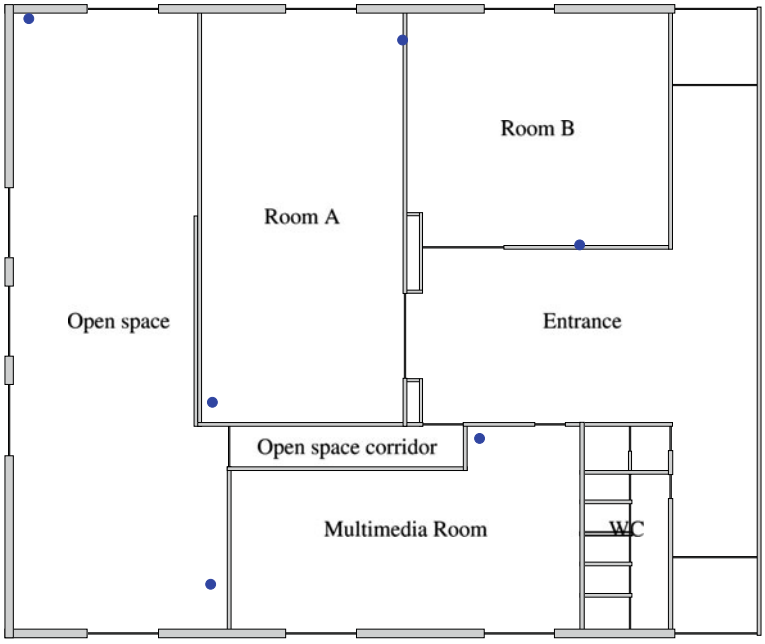


Fig. 2 Trento dataset's map: 30 m × 20 m. (courtesy of [9])

Each fingerprint is measured at a sample location by a person carrying a PDA to receive Wi-Fi signals from the access points. The PDA always points north.

A random half $Train$ of this collection (128 samples) is used for training and the other half $Test$ (129 samples) for testing purposes. Out of the training samples, we randomly create two groups of samples: the labeled group T (with the location labels intact) and the unlabeled group U (with the location labels removed). It is noted that $T, U \subset Train$ and $T \cap U = \emptyset$. The size of T is set to be 10, 20, …, or 70% of $|Train|$, and given T, the size of U is set to be 10, 20, …, or 100% of $|Train \backslash T|$. For each combination of these sizes, the average location when tested with $Test$ is averaged over 10 times running the simulation (with random generations of T and U). 1-NN is used for testing; that is, given a test fingerprint, its estimated location is the location of the nearest fingerprint in the fingerprint map.

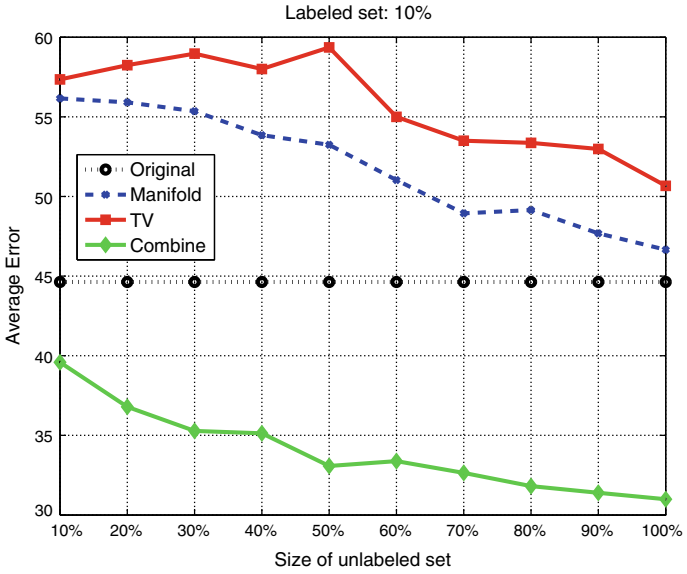
Figures 3 and 4 plot the average error for various cases of $|T|$ and $|U|$, comparing the following techniques.

- Original: Only the labeled set T is used as the training fingerprint map.
- Combine: The set $T \cup U$, where the ground-truth label is provided for every fingerprint, is used as the training fingerprint map.
- TV: The Total Variable Regularization method is used to estimate the label for U, and then, $T \cup U$ is used for training.
- Manifold: The Manifold Regularization method is used to estimate the label for U, and then, $T \cup U$ is used for training.

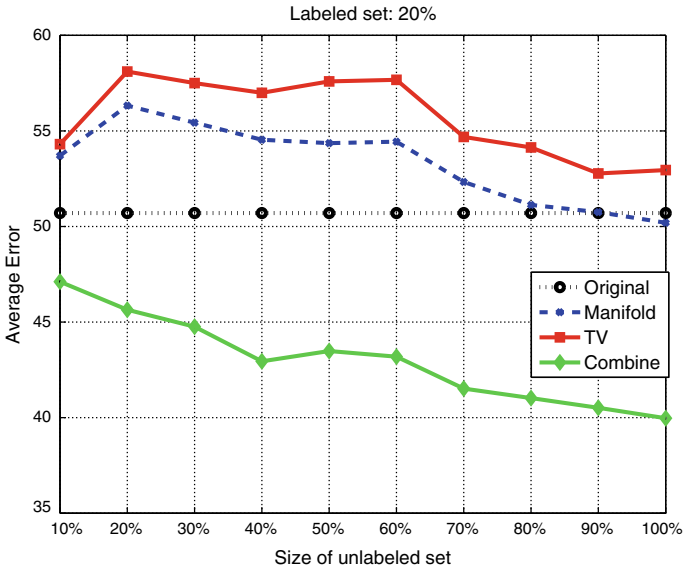
The following patterns are observed:

- Regardless of the size of the labeled set, Manifold and TV tend to be increasingly effective as the size of the unlabeled set increases.
- When the labeled set is small (e.g., 10, 20% in Fig. 3), regularization does not help. Only when the labeled set gets sufficiently large (e.g., 60, 70% in Fig. 4), we start to see its effect. This finding is understandable because a small labeled set offers too little information to be useful for the training.
- Manifold is consistently more accurate than TV. This is different from the observation in the area of image processing where TV is known to be better. This suggests that, unlike images which often have edges, the fingerprint space may not exhibit "edges" of fingerprints (i.e., a path in the fingerprint graph) located at a small cluster of locations isolated from the rest of locations. This could be due to the penetrableness of the Wi-Fi signal in the indoor area.

That said, these findings are meant to be suggestive rather than conclusive as the dataset used is small. Nevertheless, they show the potential effectiveness of both Manifold Regularization and TV Regularization in enriching the training dataset for fingerprint-based localization.

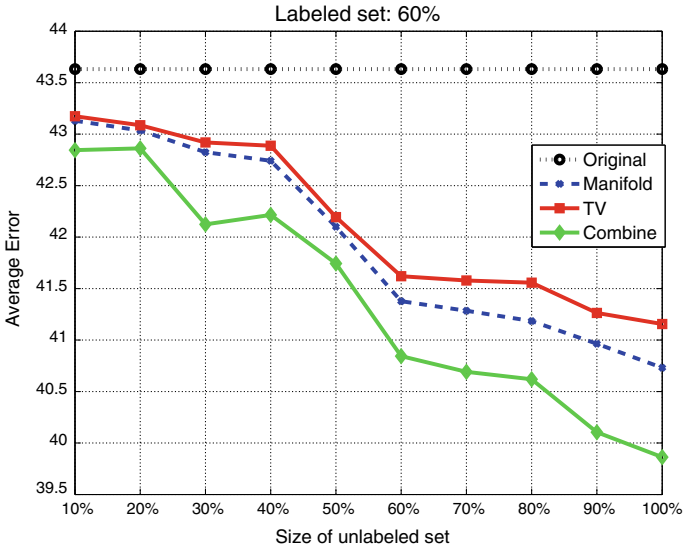


(a) $|T| = 10\%$

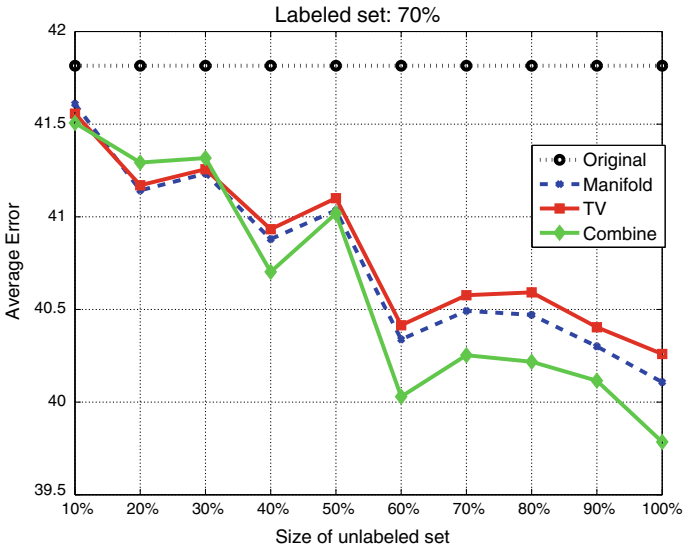


(b) $|T| = 20\%$

Fig. 3 Effectiveness of Manifold Regularization and Total Variation Regularization: showing average location error for the case of low label rate. The unit for y-axis is 0.1 m [19]



(a) $|T| = 60\%$



(b) $|T| = 70\%$

Fig. 4 Effectiveness of Manifold Regularization and Total Variation Regularization: showing average location error for the case of high label rate. The unit for y-axis is 0.1 m [19]

4 Trajectory Computation

Suppose that a mobile device is moving along a trajectory, during which we obtain a sequence of fingerprints, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, at times $1, 2, \ldots, n$, respectively. Some of them may be obtained with location information, but most without. Note that this is a sequence of fingerprints whose time order matters, not a set of fingerprints as for enriching the training data in the previous section. We need to compute the location at the current time n.

In a study on trajectory tracking, Rallapalli et al. [21] confirmed that real-world mobility of a device often exhibits its moving at a constant velocity for a long period of time before changing speed. Consequently, the quantity

$$|(f(\mathbf{x}_i) - f(\mathbf{x}_{i-1})) - (f(\mathbf{x}_{i-1}) - f(\mathbf{x}_{i-2}))| = |f(\mathbf{x}_i) + f(\mathbf{x}_{i-2}) - 2f(\mathbf{x}_{i-1})|$$

for most i should be close to zero. We refer to this property as the temporal smoothness in the fingerprint space, in contrast to the spatial smoothness discussed in the previous section.

Tran and Zhang [31] showed that temporal smoothness is more effective than spatial smoothness for trajectory construction if one property is exclusively enforced in the regularization. As an illustration, given the ground-truth trajectory of a moving device shown in Fig. 5, in all three cases where 10 or 50 or 90% of the fingerprint sequence is labeled (at random), a better trajectory estimate is obtained by enforcing temporal smoothness than by enforcing spatial smoothness; see Figs. 6, 7, and 8 for the case of 10% labeled, 50%, and 90%, respectively. In these figures, we show the location estimated for a fingerprint instantly at the time it is observed. The first point is always put at the center because in the fingerprint sequence under evaluation it happens to be unlabeled and there is no labeled fingerprint available for learning. As can be seen, in all cases of label rate, the temporally regularized trajectory resembles the ground-truth trajectory more closely than the spatially regularized trajectory does. Even in the case only 50% of the fingerprints are labeled, temporal regularization results in a trajectory (Fig. 7) comparable to the trajectory produced by spatial regularization for the case 90% labeled (Fig. 8).

Tran et al. [32] proposed a unified regularization framework combining both properties as follows:

$$\min_{f \in \mathcal{H}_K} \left\{ J(f) = \sum_{i=1}^{n} h(\mathbf{x}_i)(f(\mathbf{x}_i) - y(\mathbf{x}_i))^2 + \lambda_K \|f\|_K^2 + \lambda_S S(f) + \lambda_T T(f) \right\}, \tag{9}$$

Fig. 5 A test trajectory visiting 185 locations on the computer science department floor (68 m × 63 m) at UMass Boston

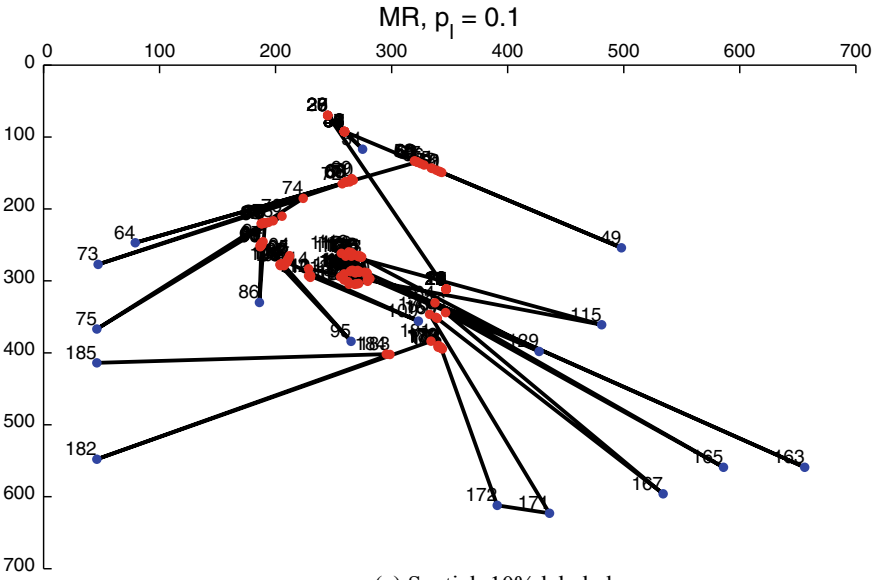
where

$$S(f) = \sum_{i,j=1}^{n} w(\mathbf{x}_i, \mathbf{x}_j)(f(\mathbf{x}_i) - f(\mathbf{x}_j))^2$$

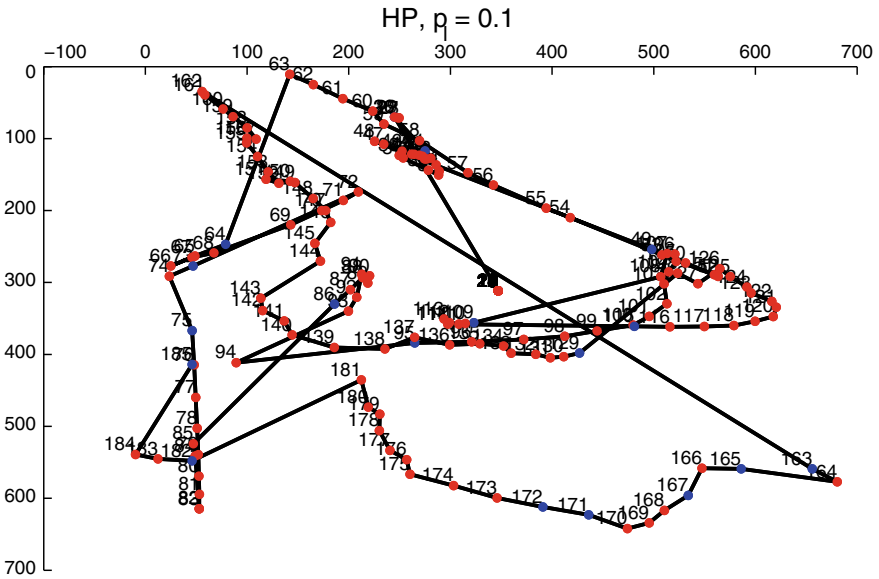
$$T(f) = \sum_{i=3}^{n} (f(\mathbf{x}_i) + f(\mathbf{x}_{i-2}) - 2f(\mathbf{x}_{i-1}))^2$$

are the regularizers to enforce spatial smoothness and temporal smoothness, respectively. This framework extends the manifold regularization framework of Belkin et al. [27] (by setting $\lambda_T = 0$) with the temporal regularizer $T(f)$. The weights λ_K, λ_S, $\lambda_T \in [0, \infty)$ are to tune the importance of the smoothness terms (kernel, spatial, temporal). Recall that $h(.)$ is the label indicator function.

Let $\mathbf{D}$ be the second-order difference matrix of size $n \times n$,

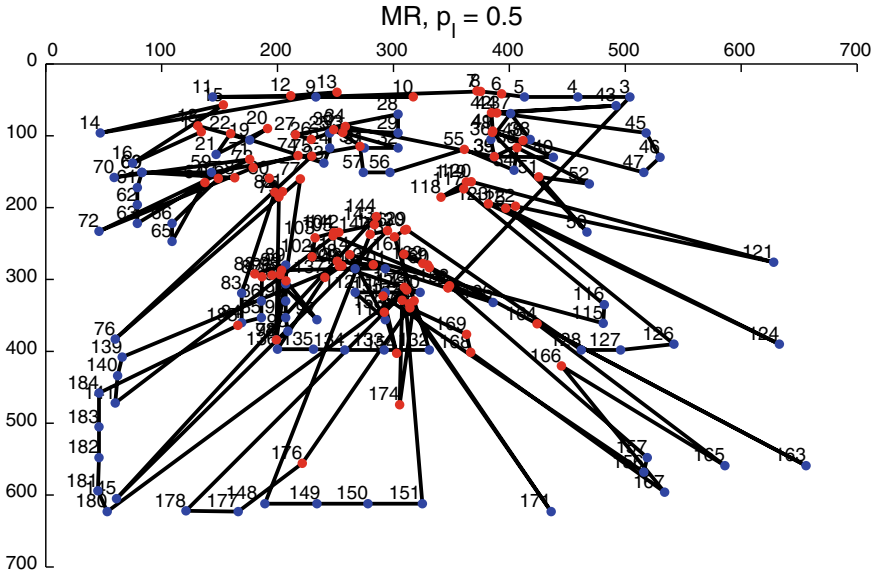


(a) Spatial: 10% labeled

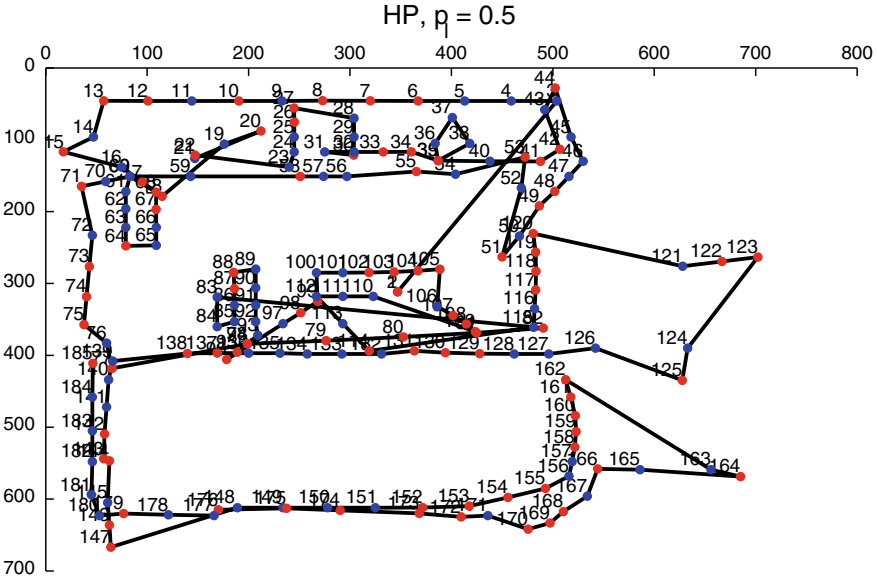


(b) Temporal: 10% labeled

Fig. 6 Effect of spatial regularization versus temporal regularization for the case 10% of the fingerprints is labeled: showing the estimated trajectory, where red-colored points are location estimates for unlabeled fingerprints and blue-colored points are the ground-truth locations of the labeled fingerprints; the numbers represent the ID of the fingerprints sorted in time of measurement. The ground-truth trajectory is shown in Fig. 5 [31]

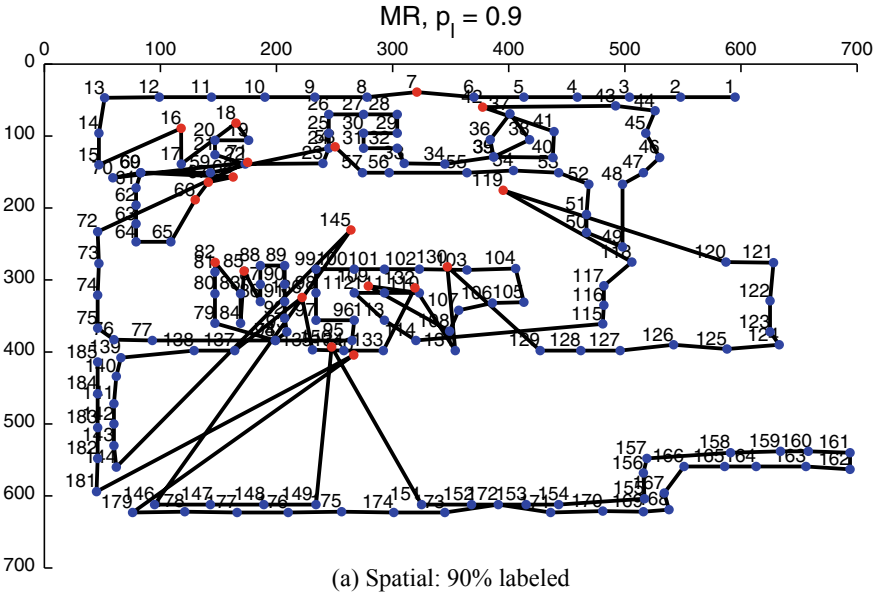


(a) Spatial: 50% labeled

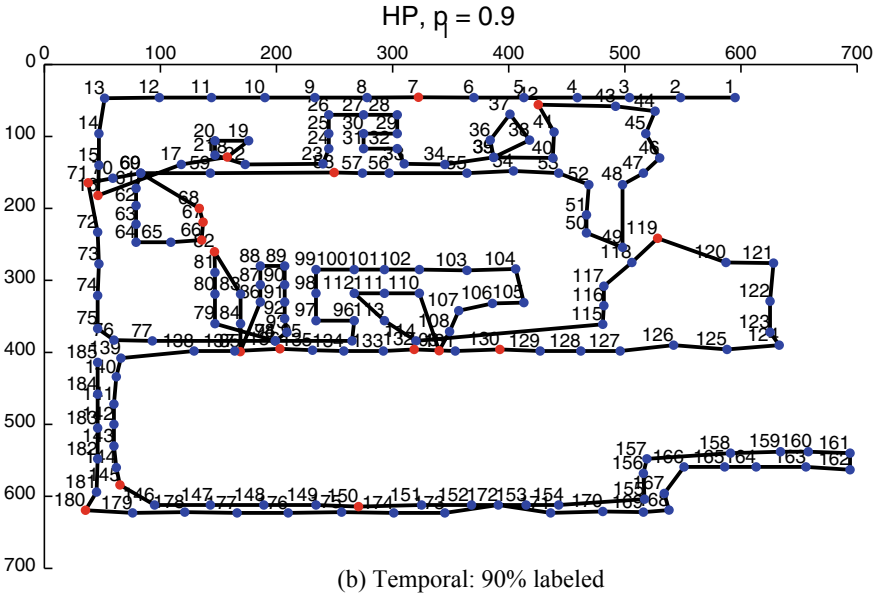


(b) Temporal: 50% labeled

Fig. 7 Effect of spatial regularization versus temporal regularization for the case 50% of the fingerprints is labeled: showing the estimated trajectory, where red-colored points are location estimates for unlabeled fingerprints and blue-colored points are the ground-truth locations of the labeled fingerprints; the numbers represent the ID of the fingerprints sorted in time of measurement. The ground-truth trajectory is shown in Fig. 5 [31]



(a) Spatial: 90% labeled



(b) Temporal: 90% labeled

Fig. 8 Effect of spatial regularization versus temporal regularization for the case 90% of the fingerprints is labeled: showing the estimated trajectory, where red-colored points are location estimates for unlabeled fingerprints and blue-colored points are the ground-truth locations of the labeled fingerprints; the numbers represent the ID of the fingerprints sorted in time of measurement. The ground-truth trajectory is shown in Fig. 5 [31]

$$\mathbf{D} = \begin{bmatrix} 0 & 0 & 0 & 0 & \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & 0 & 0 & 0 & 0 & \ldots & \ldots & \ldots & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & \ldots & \ldots & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & \ldots & \ldots & \ldots & 0 & 1 & -2 & 1 & 0 \\ 0 & \ldots & \ldots & \ldots & \ldots & 0 & 1 & -2 & 1 \end{bmatrix}_{n \times n}$$

Using the same matrix notations, $\mathbf{f}$, $\mathbf{y}$, $\mathbf{H}$, $\mathbf{L}$, and $\mathbf{K}$, as earlier, we have

$$\mathbf{Df} = \begin{bmatrix} 0 & 0 & 0 & 0 & \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & 0 & 0 & 0 & 0 & \ldots & \ldots & \ldots & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & \ldots & \ldots & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & \ldots & \ldots & \ldots & 0 & 1 & -2 & 1 & 0 \\ 0 & \ldots & \ldots & \ldots & \ldots & 0 & 1 & -2 & 1 \end{bmatrix}_{n \times n} \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \ldots \\ f(\mathbf{x}_n) \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ f(\mathbf{x}_1) - 2f(\mathbf{x}_2) + f(\mathbf{x}_3) \\ f(\mathbf{x}_2) - 2f(\mathbf{x}_3 + f(\mathbf{x}_4) \\ \ldots \\ f(\mathbf{x}_{n-2}) - 2f(\mathbf{x}_{n-1}) + f(\mathbf{x}_n) \end{bmatrix}$$

and so

$$T(f) = \sum_{i=3}^{n} (f(\mathbf{x}_i) + f(\mathbf{x}_{i-2}) - 2f(\mathbf{x}_{i-1}))^2 = (\mathbf{Df})^\intercal \mathbf{Df} = \mathbf{f}^\intercal \mathbf{D}^\intercal \mathbf{Df} = \mathbf{f}^\intercal \mathbf{Bf}$$

where

$$\mathbf{B} = \mathbf{D}^\intercal \mathbf{D} = \begin{bmatrix} 1 & -2 & 1 & 0 & \ldots & \ldots & \ldots & \ldots & 0 \\ -2 & 5 & -4 & 1 & 0 & \ldots & \ldots & \ldots & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 & \ldots & \ldots & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & -4 & 6 & -4 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & 0 & 1 & -4 & 6 & -4 & 1 \\ 0 & \ldots & \ldots & \ldots & 0 & 1 & -4 & 5 & -2 \\ 0 & \ldots & \ldots & \ldots & \ldots & 0 & 1 & -2 & 1 \end{bmatrix}_{n \times n}.$$

Also, $S(f) = \mathbf{f}^\top \mathbf{L}\mathbf{f}$. We now minimize the following regularized risk:

$$\min_{\mathbf{f}} \left\{ J(\mathbf{f}) = (\mathbf{f}-\mathbf{y})^\top \mathbf{H}(\mathbf{f}-\mathbf{y}) + \lambda_K \|\mathbf{f}\|_K^2 + \lambda_S \mathbf{f}^\top \mathbf{L}\mathbf{f} + \lambda_T \mathbf{f}^\top \mathbf{B}\mathbf{f} \right\}. \tag{10}$$

Proposition 2 *The minimizer of risk $J(\mathbf{f})$ in (10) admits the following solution*

$$\mathbf{f}^* = (\lambda_K \mathbf{I} + \mathbf{K}(\mathbf{H} + \lambda_S \mathbf{L} + \lambda_T \mathbf{B}))^{-1} \mathbf{K}\mathbf{H}\mathbf{y}. \tag{11}$$

Proof A proof can be derived in a way similar to the proof of Proposition 1. The complete proof is provided in [32].

A comparison of this framework to the nearest neighbor (NN) framework has been conducted in [32]. For example, for the same ground-truth trajectory in Fig. 5, using a sequence of 185 fingerprints with 10% labeled, the corresponding estimate resulted from each framework is visualized in Fig. 9; the regularization framework is the better. Figure 10 gives the location error comparison for other cases of label rates.

5 Online Algorithm

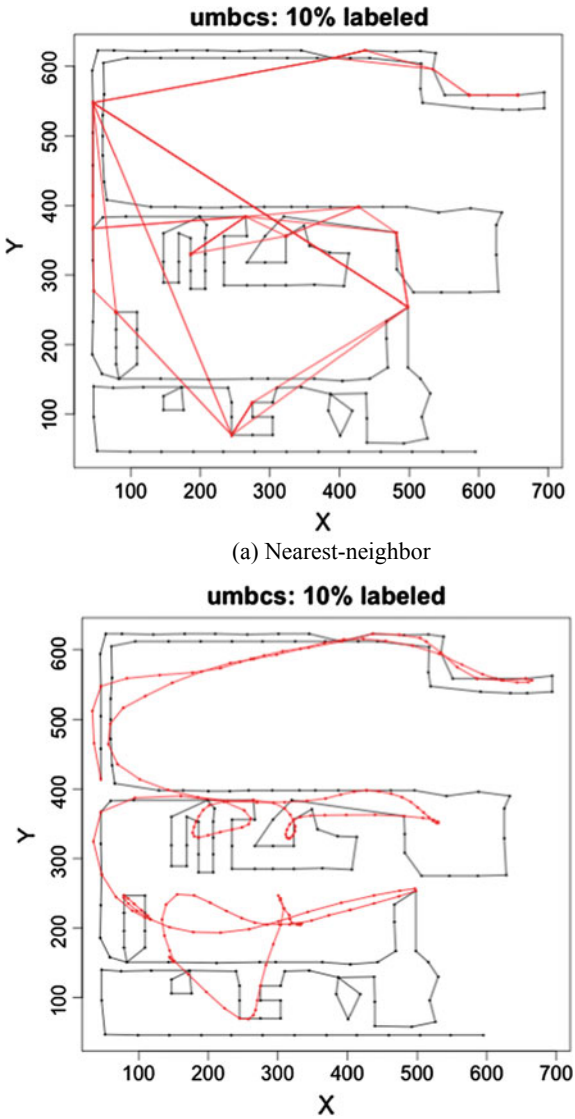
Assuming the problem setting in the previous section and using the same notations, we now discuss how the location of a moving device can be computed from its sequentially obtained fingerprints in an online manner. Note that to compute the solution $\mathbf{f}^*$ as in Eq. (11) requires a $O(n^3)$-time and $O(n^2)$-space algorithm. As such, when the fingerprint stream goes larger, it will become infeasible to compute the location of the device in real time. Fortunately, as aforementioned, the fingerprint space should be sparse in both time and space. This implies the possibility to approximate the growing set of fingerprints with only a sparse representation which we can use to estimate location in real time.

5.1 Sparse Representation

Let us represent the set of fingerprints $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ compactly as a multi-set

$$\{(\bar{\mathbf{x}}_1, m_1), (\bar{\mathbf{x}}_2, m_2), \ldots, (\bar{\mathbf{x}}_k, m_k)\}$$

where the representative elements are $\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \ldots, \bar{\mathbf{x}}_k$ and their respective multiplicities $m_1, m_2, \ldots, m_k$ ($\sum_{i=1}^{k} m_i = n$). In other words, there are m_1 fingerprints with value $\bar{\mathbf{x}}_1$, m_2 fingerprints with value $\bar{\mathbf{x}}_2$, etc. Denote the corresponding compact vector for estimated locations by



(a) Nearest-neighbor

(b) Regularization

Fig. 9 Regularization versus nearest neighbor for computation of mobile trajectory: showing the estimated trajectory (red-colored) given the same fingerprint sequence with 10% labeled; the ground-truth trajectory is black-colored. The ground-truth trajectory is shown in Fig. 5 [32]

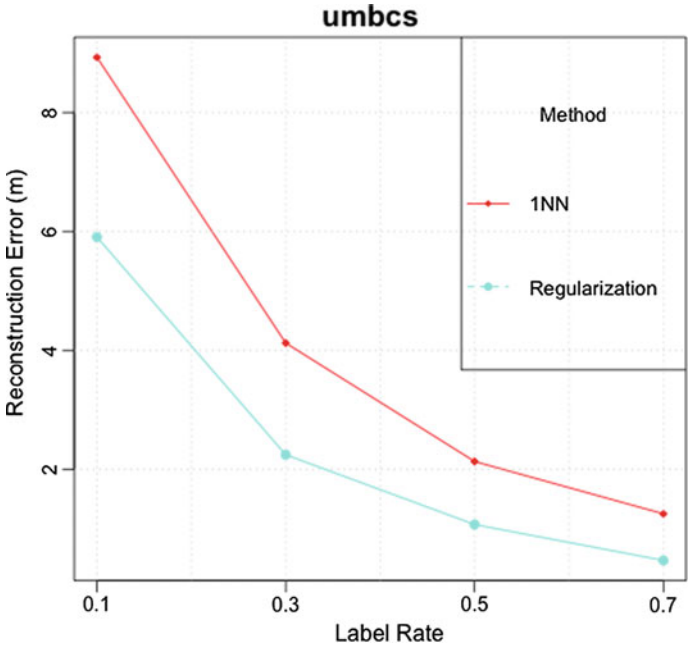


Fig. 10 Regularization versus nearest neighbor for computation of mobile trajectory: showing location error as more fingerprints are available as labeled

$$\bar{\mathbf{f}} = \begin{bmatrix} f(\bar{\mathbf{x}}_1) \\ f(\bar{\mathbf{x}}_2) \\ \dots \\ f(\bar{\mathbf{x}}_k) \end{bmatrix}.$$

Similarly, let

$$\bar{\mathbf{y}} = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \dots \\ \bar{y}_k \end{bmatrix}$$

be the compact vector for ground-truth locations. If no ground-truth information is available for fingerprint $\bar{\mathbf{x}}_i$ and neither for any fingerprint that $\bar{\mathbf{x}}_i$ represents, then $\bar{y}_i$ is set to 0; otherwise, $\bar{y}_i$ is the ground-truth location associated with $\bar{\mathbf{x}}_i$. We can transform the original minimization problem in (9) to a more compact version using only the representative fingerprints.

Since we are looking for $f \in \mathscr{H}_K$, let $f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x})$. We have

$$
\begin{aligned}
f(\bar{\mathbf{x}}_i) &= \sum_{j=1}^{n} \alpha_j K(\mathbf{x}_j, \bar{\mathbf{x}}_i) \\
&= \sum_{j=1}^{k} K(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_i) \underbrace{\sum_{p|\mathbf{x_p}=\bar{\mathbf{x}}_j} \alpha_p}_{\bar{\alpha}_j} \\
&= \sum_{j=1}^{k} \bar{\alpha}_j K(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_i),
\end{aligned}
$$

and so $\bar{\mathbf{f}} = \bar{\mathbf{K}}\bar{\boldsymbol{\alpha}}$ where $\bar{\mathbf{K}} = \left[\bar{k}_{ij} = K(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_i)\right]_{k\times k}$ and $\bar{\boldsymbol{\alpha}} = [\bar{\alpha}_1, \bar{\alpha}_2, \ldots, \bar{\alpha}_k]^\intercal$. It follows that the kernel regularization term can be expressed as $\|\mathbf{f}\|_K^2 = \bar{\boldsymbol{\alpha}}^\intercal \bar{\mathbf{K}} \bar{\boldsymbol{\alpha}}$.

Next, the estimation error with respect to the labeled fingerprints can be rewritten as follows:

$$
\sum_{i=1}^{n} h_i (f(\mathbf{x}_i) - y_i)^2 = \sum_{i=1}^{k} \bar{h}_i (f(\bar{\mathbf{x}}_i) - \bar{y}_i)^2 = (\bar{\mathbf{f}} - \bar{\mathbf{y}})^\intercal \bar{\mathbf{H}} (\bar{\mathbf{f}} - \bar{\mathbf{y}})
$$

where $\bar{\mathbf{H}} = diag(\bar{h}_1, \bar{h}_2, \ldots, \bar{h}_k)$ such that $\bar{h}_i = \sum_{j|\mathbf{x}_j=\bar{\mathbf{x}}_i} h(\mathbf{x}_j)$ which is the number of labeled fingerprints represented by $\bar{\mathbf{x}}_i$.

The spatial regularization term $S(f)$ can be rewritten as

$$
\begin{aligned}
S(f) &= \mathbf{f}^\intercal \mathbf{L} \mathbf{f} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{i} w(\mathbf{x}_i, \mathbf{x}_j)(f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 \\
&= \sum_{i=1}^{k} \sum_{j=1}^{i} \underbrace{m_i m_j w(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)}_{\bar{w}(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)} \left(f(\bar{\mathbf{x}}_i) - f(\bar{\mathbf{x}}_j)\right)^2 \\
&= \sum_{i=1}^{k} \sum_{j=1}^{i} \bar{w}(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) \left(f(\bar{\mathbf{x}}_i) - f(\bar{\mathbf{x}}_j)\right)^2 = \bar{\mathbf{f}}^\intercal \bar{\mathbf{L}} \bar{\mathbf{f}}
\end{aligned}
$$

where $\bar{\mathbf{L}}$ is the Laplacian matrix of the similarity graph with vertices $\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \ldots, \bar{\mathbf{x}}_k$, constructed as discussed earlier in Sect. 3.1, except that the edge weight function is $\bar{w}(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = m_i m_j w(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$.

The temporal regularization term $T(f)$ can be rewritten as

$$
\begin{aligned}
T(f) &= \mathbf{f}^\intercal \mathbf{B} \bar{\mathbf{f}} \\
&= \sum_{i=1}^{n} f(\mathbf{x}_i) \sum_{j=1}^{n} f(\mathbf{x}_j) b_{ij} \\
&= \sum_{i=1}^{n} f(\mathbf{x}_i) \sum_{j=1}^{k} f(\bar{\mathbf{x}}_j) \sum_{q|\mathbf{x_q}=\bar{\mathbf{x}}_j} b_{iq} \\
&= \sum_{j=1}^{k} f(\bar{\mathbf{x}}_j) \sum_{i=1}^{t} f(\mathbf{x}_i) \sum_{q|\mathbf{x_q}=\bar{\mathbf{x}}_j} b_{iq} \\
&= \sum_{j=1}^{k} f(\bar{\mathbf{x}}_j) \sum_{i=1}^{k} f(\bar{\mathbf{x}}_i) \left(\underbrace{\sum_{p|\mathbf{x_p}=\bar{\mathbf{x}}_i} \sum_{q|\mathbf{x_q}=\bar{\mathbf{x}}_j} b_{pq}}_{\bar{b}_{ij}} \right) \\
&= \sum_{i=1}^{k} \sum_{j=k}^{k} f(\bar{\mathbf{x}}_i) f(\bar{\mathbf{x}}_j) \bar{b}_{ij} = \bar{\mathbf{f}}^\intercal \bar{\mathbf{B}} \bar{\mathbf{f}}
\end{aligned}
$$

where $\bar{\mathbf{B}} = [\bar{b}_{ij}]_{k \times k}$. Intuitively, $\bar{\mathbf{B}}$ is a compact version of $\mathbf{B}$ by merging (summing up values of) all the entries in $\mathbf{B}$ whose row/column indices correspond to fingerprints having identical values.

Therefore, the risk $J(\mathbf{f})$ in (10) can be expressed as

$$
J(\mathbf{f}) = (\bar{\mathbf{f}} - \bar{\mathbf{y}})^\intercal \bar{\mathbf{H}} (\bar{\mathbf{f}} - \bar{\mathbf{y}}) + \lambda_K \bar{\boldsymbol{\alpha}}^\intercal \bar{\mathbf{K}} \bar{\boldsymbol{\alpha}} + \lambda_S \bar{\mathbf{f}}^\intercal \bar{\mathbf{L}} \bar{\mathbf{f}} + \lambda_T \bar{\mathbf{f}}^\intercal \bar{\mathbf{B}} \bar{\mathbf{f}}.
$$

meaning that the minimization problem in (10) can be reduced to an exact same minimization problem except that

- The input fingerprints are $\{\bar{\mathbf{x}}_i\}$ instead of $\{\mathbf{x}_i\}$.
- The weight function is $\bar{w}(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j) = m_i m_j w(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$.
- The matrices $\bar{\mathbf{L}}$, $\bar{\mathbf{H}}$, $\bar{\mathbf{B}}$, and $\bar{\mathbf{K}}$ are used in place of the matrices $\mathbf{L}$, $\mathbf{H}$, $\mathbf{B}$, and $\mathbf{K}$, respectively.

Applying Proposition 2 on this compact problem, we obtain the following result.

Corollary 1 *The minimizer of risk $J(\mathbf{f})$ in (10) admits the following solution $\mathbf{f}^* = [f^*(\mathbf{x}_1), f^*(\mathbf{x}_2), \ldots, f^*(\mathbf{x}_n)]$ where $f^*(\mathbf{x}_i)$ is the jth entry of vector*

$$
\bar{\mathbf{f}}^* = \left(\lambda_K \bar{\mathbf{I}} + \bar{\mathbf{K}} (\bar{\mathbf{H}} + \lambda_S \bar{\mathbf{L}} + \lambda_T \bar{\mathbf{B}})\right)^{-1} \bar{\mathbf{K}} \bar{\mathbf{H}} \bar{\mathbf{y}}. \tag{12}
$$

such that $\mathbf{x}_i = \bar{\mathbf{x}}_j$.

As a result of this corollary, instead of using a $O(n^3)$-time and $O(n^2)$-space algorithm for computing the solution $\mathbf{f}^*$ as in Eq. (11), we can obtain an exact solution by

computing $\bar{\mathbf{f}}^*$ in Eq. (12) which involves matrices of size $k \times k$, hence $O(k^3)$-time and $O(k^2)$-space complexities. Therefore, if k is small, we have a much more efficient algorithm.

In practice, however, it is rare to obtain two identical fingerprints even at the same location due to noise and so if we use the exact multi-set representation for the fingerprint sequence, the value of k can be as large as n. In what follows, we present an approximate algorithm that uses a fixed-size buffer of up to k representative fingerprints where k is set to a predefined constant.

5.2 Representative Buffer

Hereafter, we assume that k is a constant. At any time n, we maintain a buffer

$$B_n = \{(\tilde{\mathbf{x}}_i, \tilde{m}_i, \tilde{h}_i, \tilde{y}_i)\}_i$$

of up to k representative fingerprints $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_k \in \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where there is a count $\tilde{m}_i$ for $\tilde{\mathbf{x}}_i$ for the number of fingerprints represented by $\mathbf{x}_i$, and another count $\tilde{h}_i$ for the number of labeled fingerprints therein. Each fingerprint is assigned to its closest representative. Here, $\tilde{y}_i$ is the location of $\tilde{\mathbf{x}}_i$ if it is labeled and zero otherwise. We use the notation $\tilde{(\cdot)}$ to mean "approximate" to distinguish from $\bar{(\cdot)}$ which means "exact."

Natural candidates to be the representative fingerprints in the buffer are the k-centers (as in the well-known metric k-center clustering problem),

$$[\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_k] = \operatorname*{argmin}_{[\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_k]} \left(\max_{1 \le j \le n} \left(\min_{1 \le i \le k} \|\mathbf{x}_j - \tilde{\mathbf{x}}_i\| \right) \right).$$

The k-center representatives are a good compact representation of the fingerprint graph in terms of spatial representativity (unlikely for two fingerprints with similar values to be both selected centers) and temporal representativity (unlikely for two fingerprints observed in similar times to be both selected centers).

For a fingerprint stream, its k-centers can be computed, and updated incrementally upon receipt of each new fingerprint, by an incremental k-center clustering algorithm. In particular, we use Charikar et al.'s Doubling Algorithm [33]. The Doubling Algorithm guarantees that the distance between a fingerprint and its assigned representative is always less than the distance between any pair of centers. When applying the Doubling Algorithm, the details being listed in Algorithm 1, we enforce two rules: (1) when the new fingerprint needs to join an existing cluster, it prefers to be represented by a center that is labeled; see lines 6–11, and (2) in the re-clustering process when the number of centers exceeds k, labeled fingerprints are selected first to serve as center whenever needed; see line 23. More presence of labeled fingerprints

in the representative set should result in more information useful for the location prediction.

Algorithm 1: Incremental Update of Representative Buffer

```
/* run at time n after obtaining fingerprint x_n                    */
Input: fingerprint x_n, representative buffer B_{n-1} = {(x̃_i, m̃_i, ỹ_i)}_i
Output: new representative buffer B_n
if (n equals 1) then
    B_1 = {(x_1, 1, h(x_1), y(x_1))};
    Initialize R to a small value, e.g., 0.0001;
    return;
end
Find the labeled center x̃_i in B_{n-1} s.t. ||x_n − x̃_i|| < R and ||x_n − x̃_i|| is minimum;
if (that x̃_i is found) then
    /* Assign x_n to center x̃_i                                     */
    m̃_i ++;
    B_n = B_{n-1};
    return;
end
Find the unlabeled center x̃_i in B_{n-1} s.t. ||x_n − x̃_i|| < R and ||x_n − x̃_i|| is minimum;
if (that x̃_i is found) then
    /* Assign x_n to center x̃_i                                     */
    m̃_i ++;
    B_n = B_{n-1};
    return;
end
/* If we get here, x_n is too far, at least R away, from any
   center; create a new center                                      */
B_n = B_{n-1} ∪ {(x_n, 1, h(x_n), y(x_n))};
while (|B_n| > k) do
    /* Need to re-cluster                                           */
    R = 2 × R;
    B_temp = ∅;
    repeat
        Choose one center x̃_i in B_n, preferably labeled;
        B_n = B_n \ {(x̃_i, m̃_i, h̃_i, ỹ_i)};
        for (each center x̃_l in B_n s.t. ||x̃_l − x̃_i|| < R) do
            /* Collapse center x̃_l into center x̃_i                  */
            m̃_i = m̃_i + m̃_l;
            B_n = B_n \ {(x̃_l, m̃_l, h̃_l, ỹ_l)};
        end
        B_temp = B_temp ∪ {(x̃_i, m̃_i, h̃_i, ỹ_i)};
    until (B_n = ∅);
    B_n = B_temp;
    return;
end
```

It is possible that the buffer consists of $k' < k$ centers, for example, during the initial phase of the fingerprint stream or as a result of the re-clustering using the Doubling Algorithm; in this case, location estimation is computed using these k' centers. For simplicity of presentation, we assume below that the buffer has k representative fingerprints.

5.3 Location Estimation

Similar to how we define the matrices $\bar{\mathbf{H}}$, $\bar{\mathbf{L}}$, $\bar{\mathbf{B}}$, and $\bar{\mathbf{K}}$ for the exact multi-set, we have the corresponding matrices $\tilde{\mathbf{H}}$, $\tilde{\mathbf{L}}$, $\tilde{\mathbf{B}}$, and $\tilde{\mathbf{K}}$ for the representative fingerprints in B_n. When each new fingerprint is available, the buffer needs be updated and, accordingly, so do these matrices. Matrices $\tilde{\mathbf{H}}$, $\tilde{\mathbf{L}}$, and $\tilde{\mathbf{K}}$ can be computed easily after the buffer is updated, as follows:

$$\tilde{\mathbf{H}} = diag(\tilde{h}_1, \tilde{h}_2, \ldots, \tilde{h}_k)$$

$$\tilde{\mathbf{L}} = \left[\tilde{l}_{ij} = \begin{cases} -\tilde{m}_i \tilde{m}_j w(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) & \text{if } i \neq j \\ \sum_{p=1}^{k} \tilde{m}_i \tilde{m}_p w(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_p) & \text{otherwise} \end{cases} \right]_{k \times k}$$

$$\tilde{\mathbf{K}} = \left[\tilde{k}_{ij} = \exp\left(-\frac{|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j|^2}{2\gamma^2} \right) \right]_{k \times k}$$

Let "$\mapsto$" denote the assignment of a fingerprint to a representative. Matrix $\tilde{\mathbf{B}}$ is the following matrix

$$\tilde{\mathbf{B}} = \left[\tilde{b}_{ij} = \sum_{p | \mathbf{x_p} \mapsto \tilde{\mathbf{x}}_i} \sum_{q | \mathbf{x_q} \mapsto \tilde{\mathbf{x}}_j} b_{pq} \right]_{k \times k}. \tag{13}$$

The entries $\{\tilde{b}_{ij}\}_{k \times k}$ are incrementally updated during the same time as the buffer is being updated. Initially, at time zero, $\tilde{b}_{ij}$ is set to 0 for every $i, j \leq k$. Suppose that the current buffer is $\{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_k\}$ when we receive a new fingerprint $\mathbf{x}_n$. Because we use the Doubling Algorithm for updating the k-centers, the buffer update consists in two steps: (1) assign a representative to the new fingerprint and (2) re-cluster the representatives if necessary (when there are more than k of them).

In the first step, as a result of using the Doubling Algorithm, suppose that $\mathbf{x}_n$ is assigned to some representative $\tilde{\mathbf{x}}_i$ (see line 7 or line 13 in Algorithm 1). Because of this new assignment, only the values of $\tilde{b}_{ij}$ and $\tilde{b}_{ji}$ for $j = 1, 2, \ldots, k$ need to be updated such that

$$\tilde{b}_{ij} += \sum_{q|\mathbf{x_q}\mapsto\tilde{\mathbf{x}}_j} b_{nq}$$
$$\tilde{b}_{ji} += \sum_{q|\mathbf{x_q}\mapsto\tilde{\mathbf{x}}_j} b_{nq}.$$

Recall that in matrix $\mathbf{B}$ entry b_{pq} is zero everywhere except when $|p-q| \leq 2$. Therefore, $\tilde{b}_{ij}$ (and $\tilde{b}_{ji}$) can be efficiently computed as follows:

$$\tilde{b}_{ij} += \begin{cases} b_{n-1,n} & \text{if } \mathbf{x}_{n-1} \mapsto \tilde{\mathbf{x}}_j \wedge \mathbf{x}_{n-2} \not\mapsto \tilde{\mathbf{x}}_j \\ b_{n-2,n} & \text{if } \mathbf{x}_{n-1} \not\mapsto \tilde{\mathbf{x}}_j \wedge \mathbf{x}_{n-2} \mapsto \tilde{\mathbf{x}}_j \\ b_{n-1,n} + b_{n-2,n} & \text{if } \mathbf{x}_{n-1} \mapsto \tilde{\mathbf{x}}_j \wedge \mathbf{x}_{n-2} \mapsto \tilde{\mathbf{x}}_j \\ 0 & \text{otherwise.} \end{cases}$$

For this computation to be possible, we need to keep track of which fingerprints in the buffer represent the two latest fingerprints, $\mathbf{x}_{n-1}$ and $\mathbf{x}_{n-2}$.

It is also important to note that at each new step n, the entry $b_{n-1,n-1}$ changes value from 1 (value at time $n-1$) to 5 (value at time n). Similarly, $b_{n-2,n-2}$ changes value from 5 to 6, and $b_{n-1,n-2}$ and $b_{n-2,n-1}$ both from -2 to -4. The values of the other entries of matrix B are intact. Consequently, supposing that $\mathbf{x}_{n-1} \mapsto \tilde{\mathbf{x}}_{j_1}$ and $\mathbf{x}_{n-2} \mapsto \tilde{\mathbf{x}}_{j_2}$, we need to make the following updates:

$$\tilde{b}_{j_1 j_1} += 4$$
$$\tilde{b}_{j_2 j_2} += 1$$
$$\tilde{b}_{j_1 j_2} -= 2$$
$$\tilde{b}_{j_2 j_1} -= 2.$$

The time complexity for updating $\tilde{\mathbf{B}}$ in this step is $O(k)$.

In the second step, which takes place only if there are more than k representative fingerprints in the buffer, we need to re-cluster them. This step involves merging of the representatives. Each time when a representative $\tilde{\mathbf{x}}_l$ is collapsed into another representative $\tilde{\mathbf{x}}_i$ (see line 25 in Algorithm 1), we make the following update:

$$\begin{aligned} \forall\, j : 1 \leq j \leq k \wedge j \neq l : & \\ & \tilde{b}_{ij} += \tilde{b}_{jl} \\ & \tilde{b}_{ji} += \tilde{b}_{jl} \\ \forall\, j : 1 \leq j \leq k : & \\ & \tilde{b}_{lj} = 0 \\ & \tilde{b}_{jl} = 0. \end{aligned}$$

Algorithm 2: Online Algorithm

`/* run at time n after fingerprint` $\mathbf{x}_n$ `is observed */`

Input: fingerprint $\mathbf{x}_n$

Output: location of $\mathbf{x}_n$

Update buffer $B_n = \{(\tilde{\mathbf{x}}_i, \tilde{m}_i, \tilde{h}_i, \tilde{y}_i)\}_{i=1}^{k}$ according to Algorithm 1 and at the same time update matrix $\tilde{\mathbf{B}}$;

if *(* $\mathbf{x}_n$ *is labeled)* **then**

  Location of $\mathbf{x}_n$ is the given y_n;

  return;

end

Compute $\tilde{\mathbf{H}}$, $\tilde{\mathbf{L}}$, and $\tilde{\mathbf{K}}$;

Let $\tilde{\mathbf{y}} = [\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_k]^\intercal$ where $\tilde{y}_i$ is the given location of representative fingerprint $\tilde{\mathbf{x}}_i$ if it is labeled and set to zero otherwise; $\bar{\mathbf{I}}$ be the $n \times n$ identity matrix;

Compute $\tilde{\mathbf{f}} = \left(\lambda_K \bar{\mathbf{I}} + \tilde{\mathbf{K}}(\tilde{\mathbf{H}} + \lambda_S \tilde{\mathbf{L}} + \lambda_T \tilde{\mathbf{B}})\right)^{-1} \tilde{\mathbf{K}}\tilde{\mathbf{H}}\tilde{\mathbf{y}}$;

Location of $\mathbf{x}_n$ is the element in $\tilde{\mathbf{f}}$ that corresponds to the representative fingerprint of $\mathbf{x}_n$;

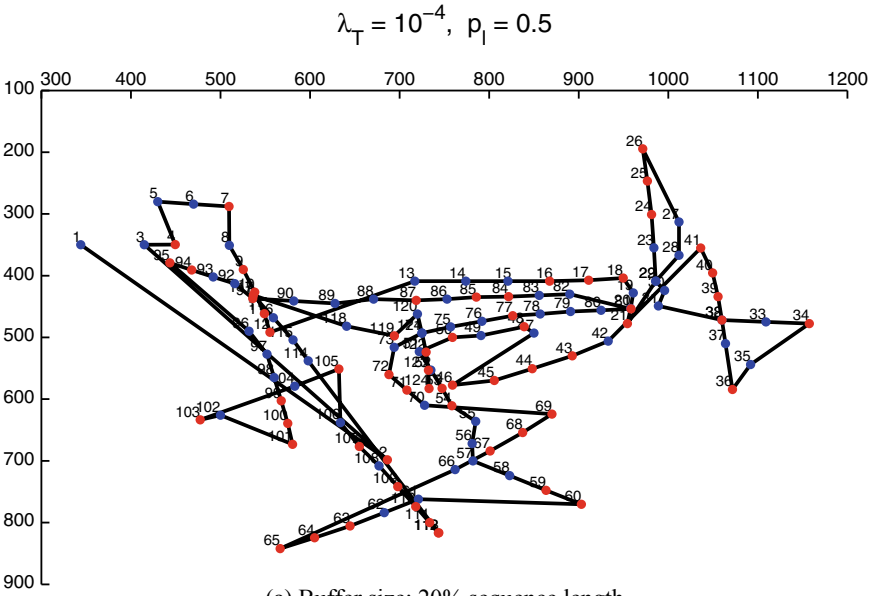
return;

The time complexity for updating $\tilde{\mathbf{B}}$ in the re-clustering step, if it occurs, is also $O(k)$.

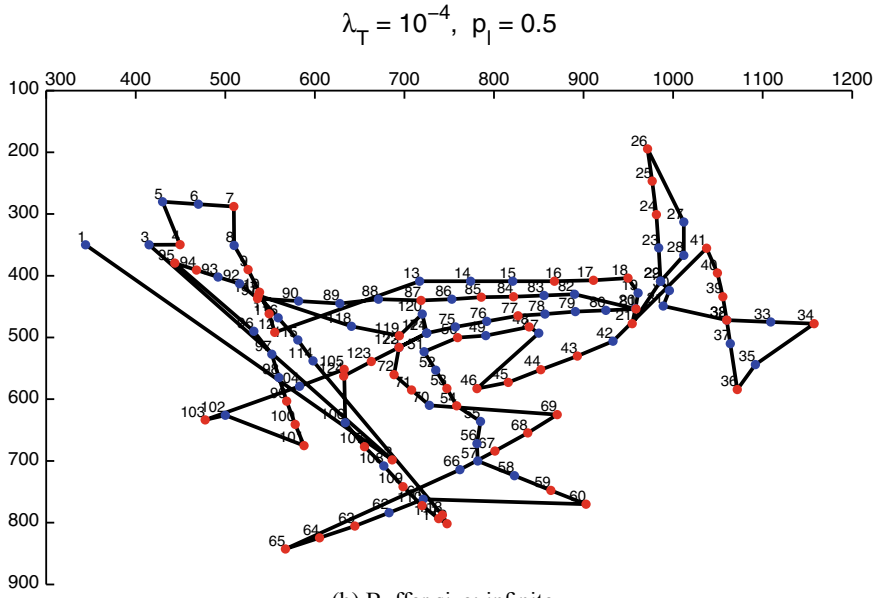
After matrices $\tilde{\mathbf{H}}$, $\tilde{\mathbf{L}}$, $\tilde{\mathbf{B}}$, and $\tilde{\mathbf{K}}$ have been updated, the location estimation is simply an application of Eq. (12) in Corollary 1 where we use $\tilde{\mathbf{H}}$, $\tilde{\mathbf{L}}$, $\tilde{\mathbf{B}}$, and $\tilde{\mathbf{K}}$ instead of $\bar{\mathbf{H}}$, $\bar{\mathbf{L}}$, $\bar{\mathbf{B}}$, and $\bar{\mathbf{K}}$. This is summarized in Algorithm 2. The total time complexity for estimating the location of each fingerprint is dominated by the time to compute the inverse matrix; hence, $O(k^3)$.

The promise of this online algorithm has been substantiated by Tran and Zhang in [34]. Figure 11 shows an example where given a sequence of 124 fingerprints, 50% of which labeled, the estimated trajectory using a buffer storing only 20% of the sequence length can closely approximate the trajectory computed using an infinite buffer. For this result, only temporal regularization is applied and the ground-truth trajectory is shown in Fig. 12; the parameters are temporal regularization coefficient $\lambda_T = 10^{-4}$, kernel regularization coefficient $\lambda_K = 10^{-6}$, and Gaussian parameters $\gamma = 1$ and $\sigma = 0.1$ for the kernel function and weight function, respectively.

In terms of computation time, Fig. 13 provides an illustration of the time advantage of the online algorithm for this trajectory. The time to locate each individual fingerprint exhibits a cubic growth until the buffer reaches its capacity (k in the online algorithm and no limit in the batch algorithm). Regardless of how large the buffer is, as long as its size is fixed, eventually the computation time will converge to a stable value, which does not increase even when the fingerprint stream gets longer. The cubic growth in time renders the batch algorithm extremely inefficient in practice where the tracking is required for an extended period of time.



(a) Buffer size: 20% sequence length



(b) Buffer size: infinite

Fig. 11 Example of the estimated trajectory for a fingerprint sequence with 50% of the fingerprints labeled. Red-colored points are location estimates for the unlabeled fingerprints, and blue-colored points are the ground-truth locations of the labeled fingerprints. The numbers represent the IDs of the fingerprints sorted in time of observation. The ground-truth trajectory is shown in Fig. 12 [34]

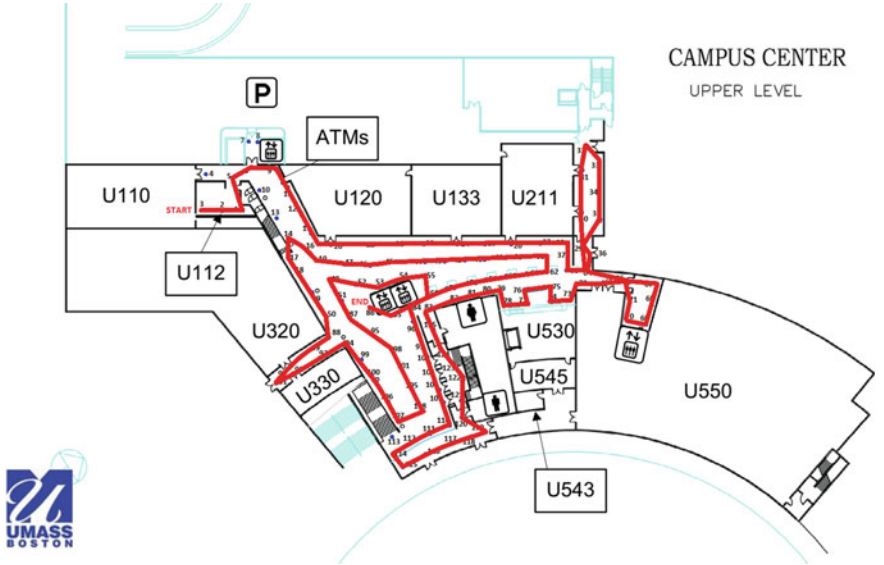


Fig. 12 A test trajectory visiting 124 locations on the upper level (185 m × 113 m) in the Campus Center at UMass Boston

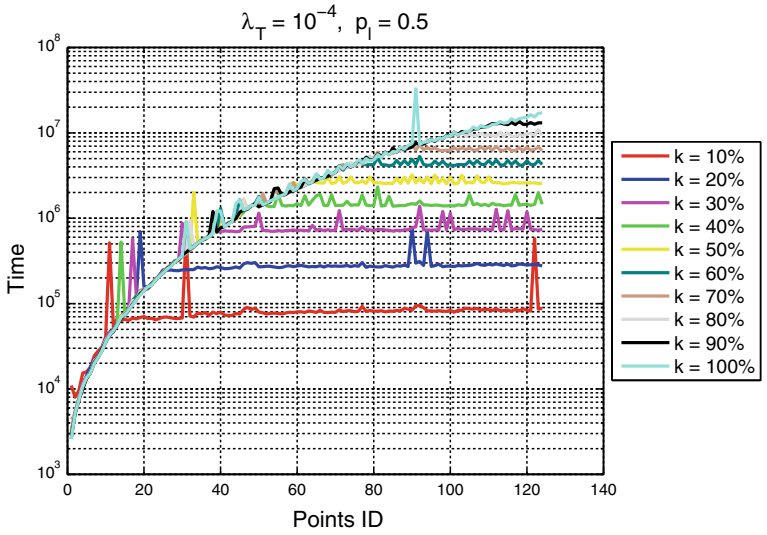


Fig. 13 Localization time (in log scale) to locate each individual fingerprint. Buffer size k is expressed as a percentage of the length of the fingerprint sequence [34]

6 Conclusions

Fingerprint-based localization and tracking can be made more effective by enriching the set of training fingerprints and by utilizing real-time fingerprints obtainable on the spot. To realize this potential, a regularization approach has been presented, which, by incorporating spatial and temporal characteristics of the fingerprint space into the regularization framework, can offer promising estimation accuracy. Also discussed is an algorithm with constant-bounded computational complexities which can be used to track the location of a moving device in real time.

Potential applications of the presented regularization frameworks are plentiful. A GPS-equipped smartphone, say, when being used in an urban shopping outlet, does not need to turn GPS on continuously; instead, it can be set to switch on once in a while and our algorithm can be used to compute the smartphone location during the GPS-free gaps. This results in great energy saving. In an indoor building, we can place location labels (e.g., RFID tags) at popular locations such as info desks, which the phone can read automatically when passing nearby; this labeled location information can be used to infer location at any other place. In tracking of autonomous underwater vehicles (AUVs) deployed in underwater environments, the AUV while submerged is tracked using a built-in inertial navigation system that has to dead reckon with GPS each time the AUV surfaces; it is desirable to improve the localization accuracy of the AUV between these GPS fixes. It is noted that in all these applications, the training data comes sequentially in real time.

Although theoretically interesting, there exist challenges when implementing a regularization framework. For example, how to find the best regularization coefficients, or in the case of online tracking, what should be the best fingerprint buffer size? Assuming a sparse fingerprint matrix, we conjecture that the best fingerprint buffer size should be logarithmic to the length of the fingerprint sequence. Another challenge is to determine the best label rate, i.e., the amount of labeled fingerprints relatively to the amount of unlabeled, as too high or too low a rate may degrade the effectiveness of the regularization. Also, in practice, there are other constraints regarding the mobility of the moving device that should be incorporated into the framework for better accuracy. Consequently, there is much more room for future research. In the mean time, the presented techniques can be used to provide benchmark for evaluation of location fingerprinting.

Acknowledgements This work was supported by the NSF award CNS-1116430. Any opinions, findings and conclusions or recommendations expressed in this material are ours and do not necessarily reflect those of the NSF.

References

1. Whitehouse, K., Karlof, C., Culler, D.: A practical evaluation of radio signal strength for ranging-based localization. SIGMOBILE Mob. Comput. Commun. Rev. **11**(1), 41–52 (2007). https://doi.org/10.1145/1234822.1234829
2. Bahl, P., Padmanabhan, V.N.: Radar: An in-building RF-based user location and tracking system. In: INFOCOM, pp. 775–784 (2000)
3. Chen, Y., Lymberopoulos, D., Liu, J., Priyantha, B.: FM-based indoor localization. In: Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, MobiSys '12, pp. 169–182. ACM, New York, NY, USA (2012). https://doi.org/10.1145/2307636.2307653
4. Hu, P., Li, L., Peng, C., Shen, G., Zhao, F.: Pharos: Enable physical analytics through visible light based indoor localization. In: Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks, HotNets-XII, pp. 5:1–5:7. ACM, New York, NY, USA (2013). https://doi.org/10.1145/2535771.2535790
5. Lorincz, K., Welsh, M.: Motetrack: a robust, decentralized approach to RF-based location tracking. Pers. Ubiquitous Comput. **11**(6), 489–503 (2007). https://doi.org/10.1007/s00779-006-0095-2
6. Yang, Z., Wu, C., Liu, Y.: Locating in fingerprint space: wireless indoor localization with little human intervention. In: Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, Mobicom '12, pp. 269–280. ACM, New York, NY, USA (2012). https://doi.org/10.1145/2348543.2348578
7. Roos, T., Myllymäki, P., Tirri, H., Misikangas, P., Sievänen, J.: A probabilistic approach to WLAN user location estimation. Int. J. Wireless Inf. Netw. **9**(3), 155–164 (2002). https://doi.org/10.1023/A:1016003126882
8. Laoudias, C., Eliades, D.G., Kemppi, P., Panayiotou, C.G., Polycarpou, M.M.: Indoor localization using neural networks with location fingerprints. In: Proceedings of the 19th International Conference on Artificial Neural Networks: Part II, ICANN '09, pp. 954–963. Springer, Berlin, Heidelberg (2009)
9. Brunato, M., Battiti, R.: Statistical learning theory for location fingerprinting in wireless lans. Comput. Netw. **47**(6), 825–845 (2005). https://doi.org/10.1016/j.comnet.2004.09.004
10. Figuera, C., Rojo-Álvarez, J.L., Wilby, M., Mora-Jiménez, I., Caamaño, A.J.: Advanced support vector machines for 802.11 indoor location. Signal Process. **92**(9), 2126–2136 (2012). https://doi.org/10.1016/j.sigpro.2012.01.026
11. Wu, C.L., Fu, L.C., Lian, F.L.: WLAN location determination in e-home via support vector classification. In: 2004 IEEE International Conference on Networking, Sensing and Control, vol. 2, pp. 1026–1031 (2004)
12. Varshavsky, A., de Lara, E., Hightower, J., LaMarca, A., Otsason, V.: GSM indoor localization. Pervasive Mob. Comput. **3**(6), 698–720 (2007). https://doi.org/10.1016/j.pmcj.2007.07.004
13. Tarzia, S.P., Dinda, P.A., Dick, R.P., Memik, G.: Indoor localization without infrastructure using the acoustic background spectrum. In: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, MobiSys '11, pp. 155–168. ACM, New York, NY, USA (2011). https://doi.org/10.1145/1999995.2000011
14. Chung, J., Donahoe, M., Schmandt, C., Kim, I.J., Razavai, P., Wiseman, M.: Indoor location sensing using geo-magnetism. In: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, MobiSys '11, pp. 141–154. ACM, New York, NY, USA (2011). https://doi.org/10.1145/1999995.2000010
15. Chapelle, O., Schlkopf, B., Zien, A.: Semi-Supervised Learning, 1st edn. The MIT Press (2010)
16. Pan, J.J., Yang, Q.: Co-localization from labeled and unlabeled data using graph laplacian. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence, pp. 2166–2171. Hyderabad, India (2007)
17. Pan, J.J., Yang, Q., Chang, H., Yeung, D.Y.: A manifold regularization approach to calibration reduction for sensor-network based tracking. In: Proceedings of the Twenty-First National Conference on Artificial Intelligence, pp. 988–993. Boston, United States (2006)

18. Pulkkinen, T., Roos, T., Myllymäki, P.: Semi-supervised learning for wlan positioning. In: Proceedings of the 21th International Conference on Artificial Neural Networks—Volume Part I, ICANN '11, pp. 355–362. Springer, Berlin, Heidelberg (2011)
19. Tran, D.A., Truong, P.: Total variation regularization for training of indoor location fingerprints. In: Proceedings of ACM MOBICOM Workshop on Mission-Oriented Wireless Sensor Networking (ACM MiseNet 2013). Miami (2013)
20. Jia Zhu, Y., Liang Deng, Z., Ji, H.: Indoor localization via l1-graph regularized semi-supervised manifold learning. J. China Univ. Posts Telecommun. **19**(5), 39–91 (2012). https://doi.org/10.1016/S1005-8885(11)60298-7
21. Rallapalli, S., Qiu, L., Zhang, Y., Chen, Y.C.: Exploiting temporal stability and low-rank structure for localization in mobile networks. In: Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking, MobiCom '10, pp. 161–172. ACM, New York, NY, USA (2010). https://doi.org/10.1145/1859995.1860015
22. Wu, C., Yang, Z., Liu, Y., Xi, W.: Will: wireless indoor localization without site survey. IEEE Trans. Parallel Distrib. Syst. **24**(4), 839–848 (2013). http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.179
23. Boukerche, A., Oliveira, H.A.B.F., Nakamura, E.F., Loureiro, A.A.F.: Vehicular ad hoc networks: A new challenge for localization-based systems. Comput. Commun. **31**(12), 2838–2849 (2008). https://doi.org/10.1016/j.comcom.2007.12.004
24. Tran, D.A., Nguyen, T.: Localization in wireless sensor networks based on support vector machines. IEEE Trans. Parallel Distrib. Syst. **19**(7), 981–994 (2008)
25. Schölkopf, B., Herbrich, R., Smola, A.J.: A generalized representer theorem. In: Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory, COLT '01/EuroCOLT '01, pp. 416–426. Springer, London, UK, UK (2001)
26. Laufer-Goldshtein, B., Talmon, R., Gannot, S.: Manifold-based Bayesian inference for semi-supervised source localization. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6335–6339 (2016). https://doi.org/10.1109/ICASSP.2016.7472896
27. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. J. Mach. Learn. Res. **7**, 2399–2434 (2006)
28. Lin, T., Xue, H., Wang, L., Zha, H.: Total variation and Euler's elastica for supervised learning. In: ICML (2012)
29. Bresson, X., Zhang, R.: TV-SVM: Total variation support vector machine for semi-supervised data classification (2012). arXiv:1210.0699
30. Elmoataz, A., Lezoray, O., Bougleux, S.: Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing. Trans. Img. Proc. **17**(7), 1047–1060 (2008). https://doi.org/10.1109/TIP.2008.924284
31. Tran, D.A., Zhang, T.: Fingerprint-based location tracking with hodrick-prescott filtering. In: Proceedings of the 7th IFIP Wireless and Mobile Networking Conference (WMNC 2014). Algarve, Portugal (2014)
32. Tran, D.A., Zhang, T., Gong, S.: A regularization framework for fingerprint-based reconstruction of mobile trajectories. Int. J. Parallel Emerg. Distrib. Syst. **31**(3), 268–279 (2016). https://doi.org/10.1080/17445760.2015.1085036
33. Charikar, M., Chekuri, C., Feder, T., Motwani, R.: Incremental clustering and dynamic information retrieval. In: Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing, STOC '97, pp. 626–635. ACM, New York, NY, USA (1997). https://doi.org/10.1145/258533.258657
34. Tran, D.A., Zhang, T.: An online algorithm for fingerprint-based location tracking. In: Proceedings of the 11th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2014). Philadelphia, PA (2014)

Sense-Through-Foliage Target Detection Based on UWB Radar Sensor Networks

Qilian Liang

Abstract In this chapter, we study sense-through-foliage target detection using ultra-wideband (UWB) radar. We propose a discrete cosine transform (DCT)-based approach for sense-through-foliage target detection when the echo signal quality is good, and a radar sensor network (RSN) and DCT-based approach when the echo signal quality is poor. A RAKE structure which can combine the echoes from different cluster members is proposed for clusterhead in the RSN. We compared our approach with the ideal case when both echoes are available, i.e., echoes with target and without target. We also compared our approach against the scheme in which 2-D image was created via adding voltages with the appropriate time offset as well as the matched filter-based approach. We observed that the matched filter-based could not work well because the UWB channel has memory. Simulation results show that our DCT-based scheme works much better than the existing approaches, and our RSN- and DCT-based approach can be used for target detection successfully while even the ideal case fails to do it. We also apply human-inspired mechanism to sense-through-foliage target detection. One of the great mysteries of the brain is cognitive control. How can the interactions between millions of neurons result in behavior that is coordinated and appears willful and voluntary? There is consensus that it depends on the prefrontal cortex (PFC). Many PFC areas receive converging inputs from at least two sensory modalities. Inspired by human's innate ability to process and integrate information from disparate, network-based sources, we apply human-inspired information integration mechanisms to target detection in cognitive radar sensor network. Humans' information integration mechanisms have been modeled using maximum-likelihood estimation (MLE) or soft-max approaches. In this chapter, we apply these two algorithms to cognitive radar sensor networks target detection. DCT is used to process the integrated data from MLE or soft-max. We apply fuzzy logic system (FLS) to automatic target detection based on the AC power values from DCT. Simulation results show that our MLE–DCT–FLS and soft-max–DCT–FLS approaches perform very well in the radar sensor network target detection.

Q. Liang (✉)
Department of Electrical Engineering, University of Texas at Arlington, Arlington, TX 76019-0016, USA
e-mail: liang@uta.edu

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_11

1 Introduction and Motivation

UWB radars are used nowadays for different applications such as subsurface sensing, classification of aircraft, collision avoidance. In all of these applications, the ultra-high resolution of UWB radars is essentially used [59]. UWB radar emissions are at a relatively low frequency typically between 100 MHz and 3 GHz. Additionally, the fractional bandwidth of the signal is very large (greater than 0.25). In this definition, bandwidth means the difference between the highest and lowest frequencies of interest and contains about 95% of the signal power [50, 51]. Such radar sensor has exceptional range resolution that also has an ability to penetrate many common materials (e.g., walls). Law enforcement personnel have used UWB ground penetrating radars (GPRs) for at least a decade. In this chapter, we will study sense-through-foliage target detection using UWB radars.

Like the GPR, sense-through-foliage radar takes advantage of UWB's very fine resolution (time gating) and low frequency of operation. In [18], Kapoor et al. studied the detection of targets obscured by a forest canopy using a UWB radar. They observed that the forest clutter observed in the radar imagery is a highly impulsive random process that is more accurately modeled with the alpha-stable processes as compared with Gaussian, Weibull, and K-distribution models. With this more accurate model, segmentation was performed on the imagery into forest and clear regions. Further, a region-adaptive symmetric alpha-stable ($S\alpha S$) constant false-alarm rate (CFAR) detector was introduced and its performance is compared with the Weibull and Gaussian CFAR detectors. The approach in [18] is a statistical model-based approach. In this chapter, we are interested in a non-statistical model-based approach for UWB sense-through-foliage target detection, and we will apply our expertise in signal processing, data fusion, sensor networks, etc. to achieve effective sense-through-foliage technology. In [32], sense-through-wall human detection was studied based on UWB radar sensors using standard deviation approach. In [34], multistep information fusion was applied to sense-through foliage target detection, and information theoretical approach [24] was applied to opportunistic sensing in sense-through foliage target detection. In [23], some preliminary work on sense-through-foliage based on discrete cosine transform was proposed. In this chapter, we are interested in investigating more features from sense-through-foliage signals and extracting as much information as possible for data fusion in radar sensor networks.

The data fusion in radar sensor networks needs waveform diversity combining. Most existing works on waveform design and selection are focused on single radar or sonar system. In 1974, Fitzgerald [16] demonstrated the inappropriateness of selection of waveforms based on measurement quality alone: The interaction between the measurement and the track can be indirect, but must be accounted for. Since then, extensive works on waveform design have been reported. Bell [4] used information theory to design radar waveforms for the measurement of extended radar targets exhibiting resonance phenomena. In [3], the singularity expansion method was used to design discriminant waveforms such as K-pulse, E-pulse, and S-pulse. Sowelam and Tewfik [47] developed a signal selection strategy for radar target classification,

and a sequential classification procedure was proposed to minimize the average number of necessary signal transmissions. Intelligent waveform selection was studied in [2, 17], but the effect of Doppler shift was not considered. In [39], time-frequency-based generalized chirps were used as waveform for detection and estimation. In [37], the performance of constant frequency (CF) and linear frequency modulated (LFM) waveform fusion from the standpoint of the whole system was studied, but the effect of clutter was not considered. In [48], a new time-frequency signal decomposition algorithm based on the S-method was proposed and evaluated on the high-frequency surface-wave radar (HFSWR) data, and demonstrated that it provided an effective way for analyzing and detecting maneuvering air targets with significant velocity changes, including target signal separation from the heavy clutter. In [49], CF and LFM waveforms were studied for a sonar system, but it was assumed that the sensor is non-intelligent (i.e., waveform cannot be selected adaptively). All the above studies and design methods focused on the waveform design or selection for a single active radar or sonar system. In [60–64], some cognitive radio-based approach was proposed for waveform design. In [45], cross-correlation properties of two radars were briefly mentioned and the binary-coded pulses using simulated annealing [9] are highlighted. However, the cross-correlation of two binary sequences such as binary-coded pulses (e.g., Barker sequence) is much easier to study than that of two analog radar waveforms. In [31], waveform design and diversity was studied with clear performance gain. In this chapter, we focus on the waveform diversity and design for radar sensor networks, as well as information fusion for target detection.

In radar sensor network (RSN), interferences among radars can be effectively reduced when waveforms are properly designed. In [25, 56], we firstly performed some theoretical studies on coexistence of phase-coded waveforms in RSN. Then, we gave the definition of new kind of triphase-coded waveforms optimized punctured zero correlation zone sequence pair set (ZCZPS) and analyze their properties, especially their optimized cross- correlation property of any two sequence pairs in the set. Besides, we applied our newly provided triphase-coded waveforms and equal gain combination technique to the system simulation and study the performances versus different number of radars in RSN under the condition of either Doppler shift or time delay. Simulation results showed that detection performances of RSN (applying our optimized punctured ZCZPS and equal gain combination) with or without Doppler shift are superior to those of single radar. In [21], similarly to the RSN, we extended our design to underwater sonar sensor networks. Sensing-through-wall will benefit various applications such as emergence rescues and military operations. In order to add more signal processing functionality, it is vital to understand the characterization of sense-through-wall channel. In [28], we proposed a statistical channel model on a basis of real experimental data using UWB noise radar. We employ CLEAN algorithm to obtain the multipath channel impulse response (CIR) and observe that the amplitude of channel coefficient at each path can be accurately characterized as T location-scale distribution. We also analyzed that the multipath contributions arrive at the receiver are grouped into clusters. The time of arrival of the clusters can be modeled as a Poisson arrival process, while within each cluster, subsequent multipath contributions or rays also arrive according to a Poisson process. However,

these arrival rates are much smaller than those of indoor UWB channels. Inspired by recent advances in MIMO radar, in [58], we introduced orthogonal pulse compression codes to MIMO radar system in order to gain better target direction finding performance. We proposed the concept and the design methodology for the optimized triphase-coded waveforms that is the optimized punctured zero correlation zone (ZCZ) sequence pair set (ZCZPS). The method is to use the optimized punctured sequence pair along with Hadamard matrix in the ZCZ. According to codes property analysis, our proposed phase- coded waveforms could provide optimized autocorrelation and cross- correlation properties in ZCZ. Then, we presented a generalized MIMO radar system model using our proposed codes and simulates the target direction finding performance in the system. The simulation results showed that diversity gain could be obtained using our orthogonal pulse compression codes for MIMO radar system. The more the antennas used, the better target direction finding performance provided. Based on the zero correlation zone (ZCZ) concept, in [29] we presented the definition and properties of a set of new triphase- coded waveforms ZCZ sequence pair set (ZCZPS) and proposed a method to use the optimized punctured sequence pair along with Hadamard matrix in the zero correlation zone in order to construct the optimized punctured ZCZ sequence pair set (optimized punctured ZCZPS). According to property analysis, the optimized punctured ZCZPS has good autocorrelation and cross-correlation properties when Doppler shift is not large. We apply it to radar target detection. The simulation results showed that optimized punctured ZCZ sequence pairs (optimized punctured ZCZPs) outperform other conventional pulse compression codes, such as the well-known polyphase code.

In [57], we presented new developed triphase code in punctured binary sequence pair. The definitions and the autocorrelation properties of the proposed code are given. Doppler shift performance is also investigated. The significant advantages of punctured binary sequence pair over conventional pulse compression codes, such as the widely used Barker codes, are zero autocorrelation sidelobes and the longer length of the code which can be as long as 31 so far. Applying the codes in the radar target detection system simulation, punctured binary sequence pair also outperforms other conventional pulse compression codes. Therefore, our proposed code can be used as good candidates for pulse compression code. In [20, 21], we applied biologically inspired approach to RSN-based target detection and used a fuzzy logic system [22] to make final decision. In [27], we performed a number of theoretical studies on constant frequency (CF) pulse waveform design and diversity in radar sensor networks (RSNs): (1) the conditions for waveform coexistence, (2) interferences among waveforms in RSN, (3) waveform diversity combining in RSN. As an application example, we apply the waveform design and diversity to automatic target recognition (ATR) in RSN and propose maximum-likelihood (ML)-ATR algorithms for non-fluctuating target as well as fluctuating target. Simulation results show that our waveform diversity-based ML-ATR algorithm performs much better than single-waveform ML-ATR algorithm for non-fluctuating targets or fluctuating targets. Conclusions are drawn based on our analysis and simulations. In [26], we designed a network of distributed radar sensors that work in an ad hoc fashion, but are grouped together by an intelligent clusterhead. This system is named radar sensor network

(RSN). A RSN not only provides spatial resilience for target detection and tracking compared to traditional radars, but also alleviates inherent radar defects such as the blind speed problem. This interdisciplinary area offers a new paradigm for parallel and distributed sensor research. We propose both coherent and noncoherent RSN detection systems applying selection combination algorithm (SCA) performed by clusterhead to take the advantage of spatial diversity. Monte Carlo simulations show that proposed RSN can provide much better detection performance than that of single radar sensor for fluctuating targets, in terms of probability of false alarm and miss detection. We also analyze the impact of Doppler shift on both coherent and noncoherent RSN detection systems at the presence of clutter. The result is that the coherent system is more robust to the noncoherent RSN. In [34], we proposed a multistep information fusion scheme for target detection through foliage and wall, using ultra-wideband (UWB) radar sensor networks. We apply an information theory to detect target with poor signal quality in dynamic forest environment. This method is motivated by the fact that echoes from the stationary target that is obscured by foliage have strong random characteristics. This is resolved by three steps of information fusion. For the first step of information fusion, we use Kullback–Leibler divergence-based weighting and generated a modified histogram. In the second step, we use entropy- and mutual information-based information fusion. Finally, we use three different fusion methods: (1) Dempster and Shafer theory of evidence; (2) proportional conflict redistribution rule 5; and (3) Bayesian network for decision fusion. Results show that when echoes are in poor quality, accurate detection can be achieved by applying our method. To demonstrate that our algorithm could be applied to other scenarios, we apply it to sense-through-wall human detection using different UWB radars, and simulation results show that our approach works well. Recently, we applied cylindrical arrays to target detection [53–55].

The rest of this chapter is organized as follows. In Sect. 2, we summarize the measurement and collection of data we used in this chapter. In Sect. 3, we propose a discrete cosine transform (DCT)-based approach for sense-through-foliage target detection with good signal quality. In Sect. 4, we propose the theory on waveform design and diversity for radar sensor networks. In Sect. 5, we propose a radar sensor network (RSN) and DCT-based approach for sense-through-foliage target detection when the signal quality is poor. In Sect. 6, human-inspired target recognition for sense-through-foliage is proposed. Section 7, fuzzy logic system is applied to automatic target detection. Section 8 concludes this chapter and discusses some future research topics.

2 Sense-Through-Foliage Data Measurement and Collection

The experiments were performed by virtual machines company supported by Air Force Research Laboratory via FOPEN Phase 2 Field Test project, and the measurements were taken on the grounds of Virtual Machines Company in Holliston,

Massachusetts [8]. The foliage experiment was constructed on a seven-ton man lift, which had a total lifting capacity of 450 kg. The limit of the lifting capacity was reached during the experiment as essentially the entire measuring apparatus was placed on the lift. The principle pieces of equipment secured on the lift are listed below:

- Dual Antenna mounting stand
- Two antennas
- Rack system (2)
- Barth pulser
- Tektronix model 7704 B oscilloscope
- IBM laptop
- HP signal generator
- Custom RF switch and power supply
- Weather shield (small hut)

Figure 1 shows the experiment under a weather shield that was constructed on the lift. The weather shield was needed to protect the equipment hoisted up with the lift. A negative side effect of this weather shield was to provide a significant sail area at the maximum lever arm relative to the lift stabilizing jacks on the ground. Lift stabilization was achieved using cables and anchor points. A system of four tethers was used under gusty conditions. The transmit and receive rotating platform systems were built using heavy gauge Unistruts, thrust bearings, and roller bearings for the multiple axes of freedom. The importance of the rigidity of the antenna mounts and the axis of rotation was in the establishment and maintenance of the antenna alignment during the measurement.

The experimental target was a trihedral reflector with a slant length of 1.5 m (as shown in Fig. 2). Throughout this work, a Barth pulse source (Barth Electronics, Inc. model 732 GL) was used. The pulse generator uses a coaxial reed switch to discharge a charge line for a very fast rise time pulse outputs. The model 732 pulse generator provides pulses of less than 50 picoseconds (ps) rise time, with amplitude from 150 V to greater than 2 KV into any load impedance through a 50 Ω coaxial line. The generator is capable of producing pulses with a minimum width of 750 ps and a maximum of 1 μs. This output pulse width is determined by charge line length for rectangular pulses, or by capacitors for 1/e decay pulses. The data collections were extensive. Twenty different positions were used, and 35 independent collections were performed at each position.

For the data we used in this chapter, each sample is spaced at 50 ps interval, and 16,000 samples were collected for each collection for a total time duration of 0.8 μs at a rate of approximately 20 Hz. We considered two sets of data from this experiment. Initially, the Barth pulse source was operated at only 1 KW peak power and the system was not sufficiently loaded for repeatable charge control pulse to pulse. Significant pulse-to-pulse variability was noted for these collections. In this set of experiments, 35 pulses reflected signal were averaged for each collection. The scheme for the sense-through-foliage target detection with "poor" signal quality will be presented in Sect. 5.

Fig. 1 This figure shows the lift with the experiment. The antennas are at the far end of the lift from the viewer under the roof that was built to shield the equipment from the elements. This picture was taken in September with the foliage largely still present. The cables coming from the lift are a ground cable to an earth ground and one of four tethers used in windy conditions

Fig. 2 The target (a trihedral reflector) is shown on the stand at 300 ft from the lift

This problem was remedied by running the pulser at higher power while protecting the radiating antenna using a non-distorting attenuator Barth 3 dB attenuator model number 142-NMFP-3. Pulse production stability was very important to this measurement effort. Pulse-to-pulse differences, if any were observed, should be due to changes in the foliage or changes in the transmitter–receiver positions relative to the foliage and target. When operated at the higher amplitudes, it was noted that the pulse source was very stable. In this set of experiments, 100 pulses reflected signals were averaged for each collection to average the variation because of the movement of foliage. The scheme for target detection with "good" signal quality will be presented in Sect. 3.

3 Sense-Through-Foliage Target Detection with Good Signal Quality: A DCT-Based Approach

In Fig. 3, we plot two collections with good signal quality, one without target on range (Fig. 3a) and the other one with target on range (Fig. 3b and target appears at around sample 13,900). To make it more clear to the readers, we provide expanded views of

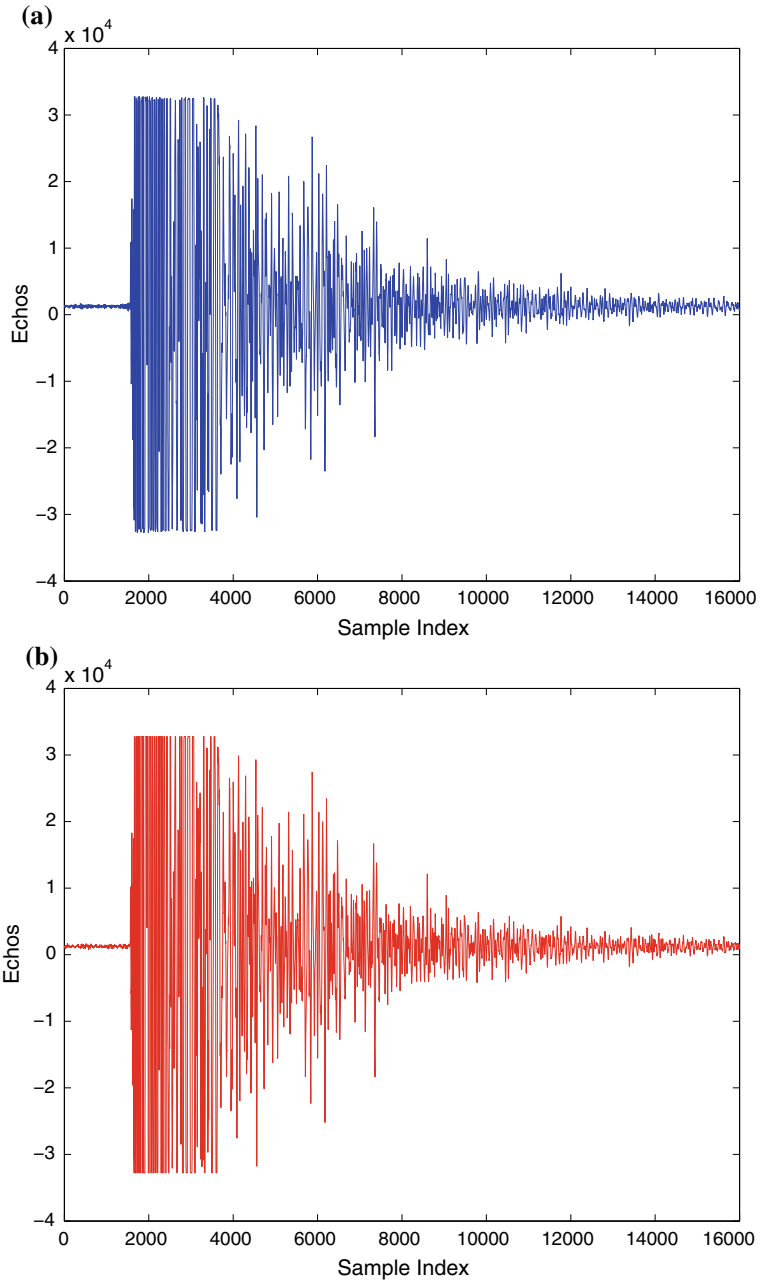


Fig. 3 Measurement with very good signal quality and 100 pulses average. **a** No target on range, **b** with target on range (target appears at around sample 13,900)

traces (with target) from sample 13,001 to 15,000 for the above two collections in Fig. 4a, b. Since there is no target in Fig. 4a, it can be treated as the response of foliage clutter. It is quite straightfoward that the target response will be the echo difference between Fig. 4a, b, which is plotted in Fig. 4c. However, it is impossible to obtain Fig. 4a (clutter echo) in practical situation if there is target on range. The challenge is how to make target detection based on Fig. 4b (with target) or Fig. 4a (no target) only?

Observe Fig. 4b, for samples where target appears (around sample 13,900), the sample strength changes much abruptly than that in Fig. 4a, which means echo from target contains more AC values than that without target. Motivated by this, we applied discrete cosine transform (DCT) to the echoes $x(iM+n)$ $(n=0,1,2,\ldots,N-1)$ where N is the DCT window length, M is the step size of each DCT window, and i is the window index. Let $x(n,i) \triangleq x(iM+n)$

$$X(K,i) = \sum_{n=0}^{N-1} x(n,i)\cos(\frac{2\pi}{N}nK) \tag{1}$$

then we cumulate the power of AC values (for $K > 2$)

$$P(i) = \sum_{K=3}^{N-1} X(K,i)^2 \tag{2}$$

For $N = 100$ and $M = 10$, we plot the power of AC values $P(i)$ versus iM (time domain sample index) in Fig. 5a, b for the above data sets in Fig. 4a, b respectively. Observe that in Fig. 5b, the power of AC values (around sample 13,900) where the target is located is non-fluctuating (monotonically increase then decrease). Although some other samples also have very high AC power values, it is very clear that they are quite fluctuating and the power of AC values behave like random noise because generally the clutter has Gaussian distribution in the frequency domain [1]. Based on our simulations, the window length N in DCT affects the performance of the target detection. The appropriate N should be the length of target impulse response with strong signal strength (see Fig. 4c). This depends on target size, UWB signal resolution, and propagation environment.

We compared our DCT-based approach to the scheme proposed in [52]. In [52], 2-D image was created via adding voltages with the appropriate time offset. In Fig. 6a, b, we plot the 2-D image created based on the above two data sets (from samples 13,800 to 14,200). However, it is not clear which image shows there is target on range.

We also compared our approach to the matched filter approach. The matched filter is by definition a filter in the radar receiver designed to maximize the SNR at its output. The impulse response of the filter having this property turns out to be a replica of the transmitted waveform's modulation function that has been reversed in time and conjugated [43]. Assume the transmitted waveform is $s(t)$, then the matched filter

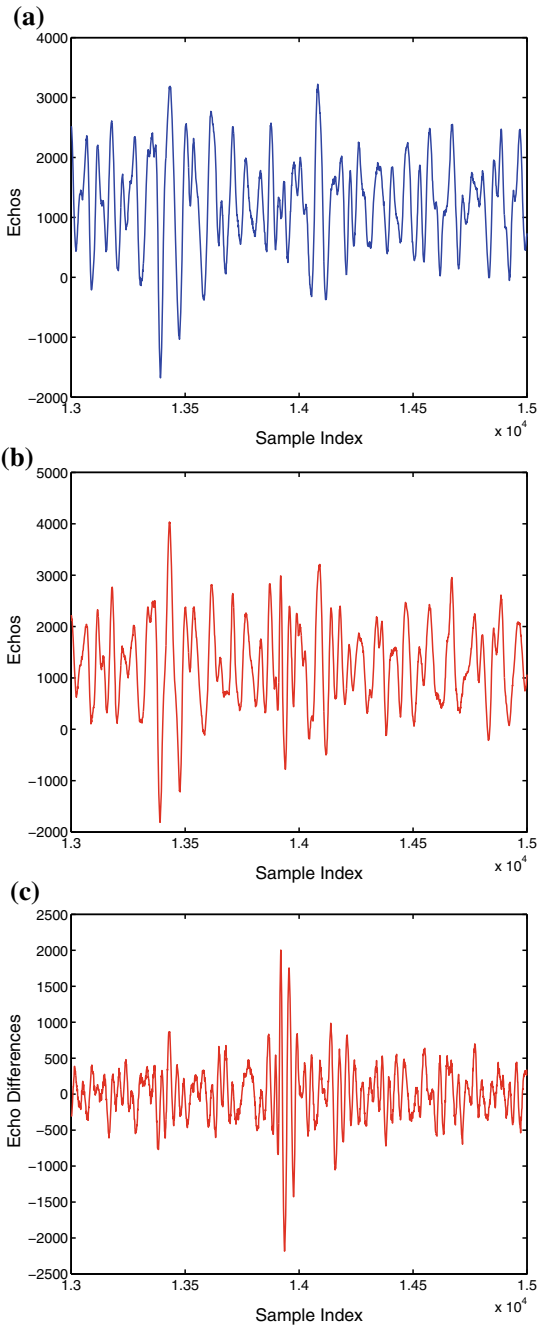


Fig. 4 Measurement with very good signal quality and 100 pulses average. **a** Expanded view of traces (with target) from samples 13,001 to 15,000. **b** Expanded view of traces (without target) from samples 13,001 to 15,000. **c** Echo differences between (**a**) and (**b**)

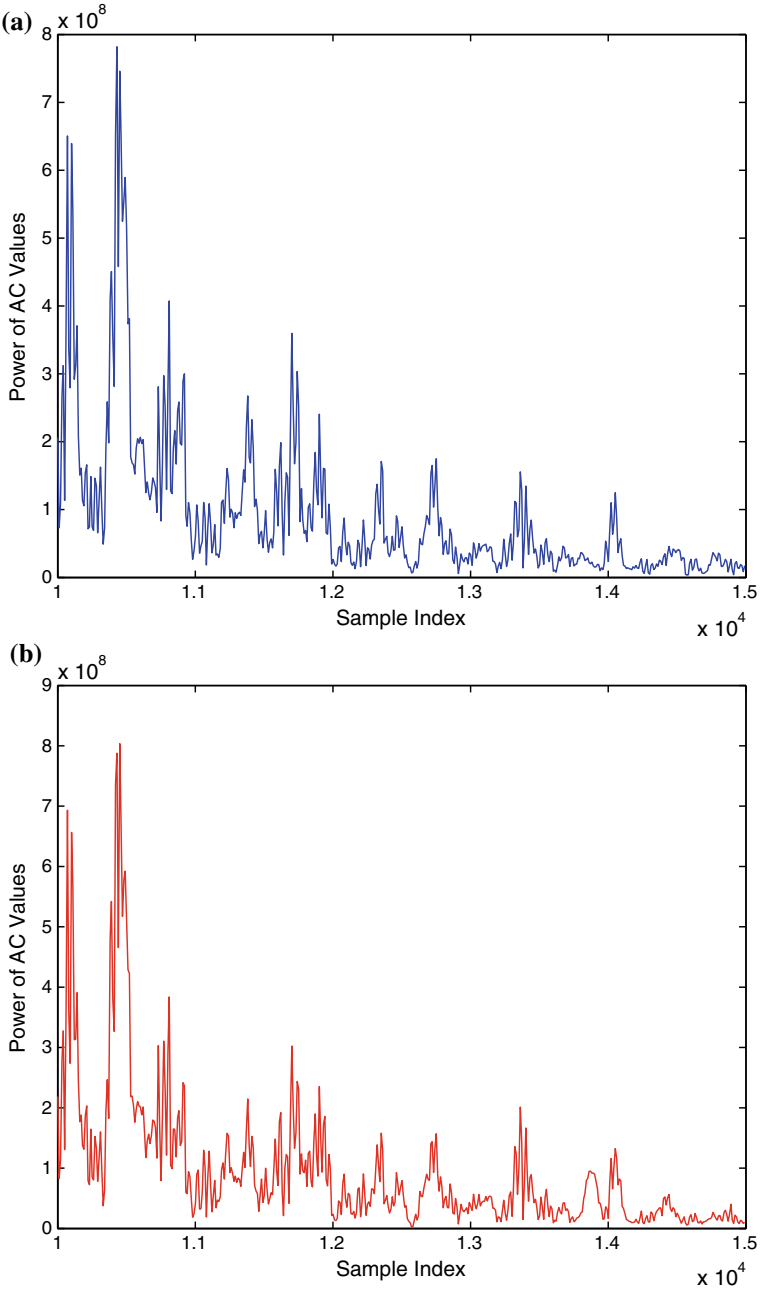


Fig. 5 The power of AC values versus sample index. **a** No target. **b** With target in the field

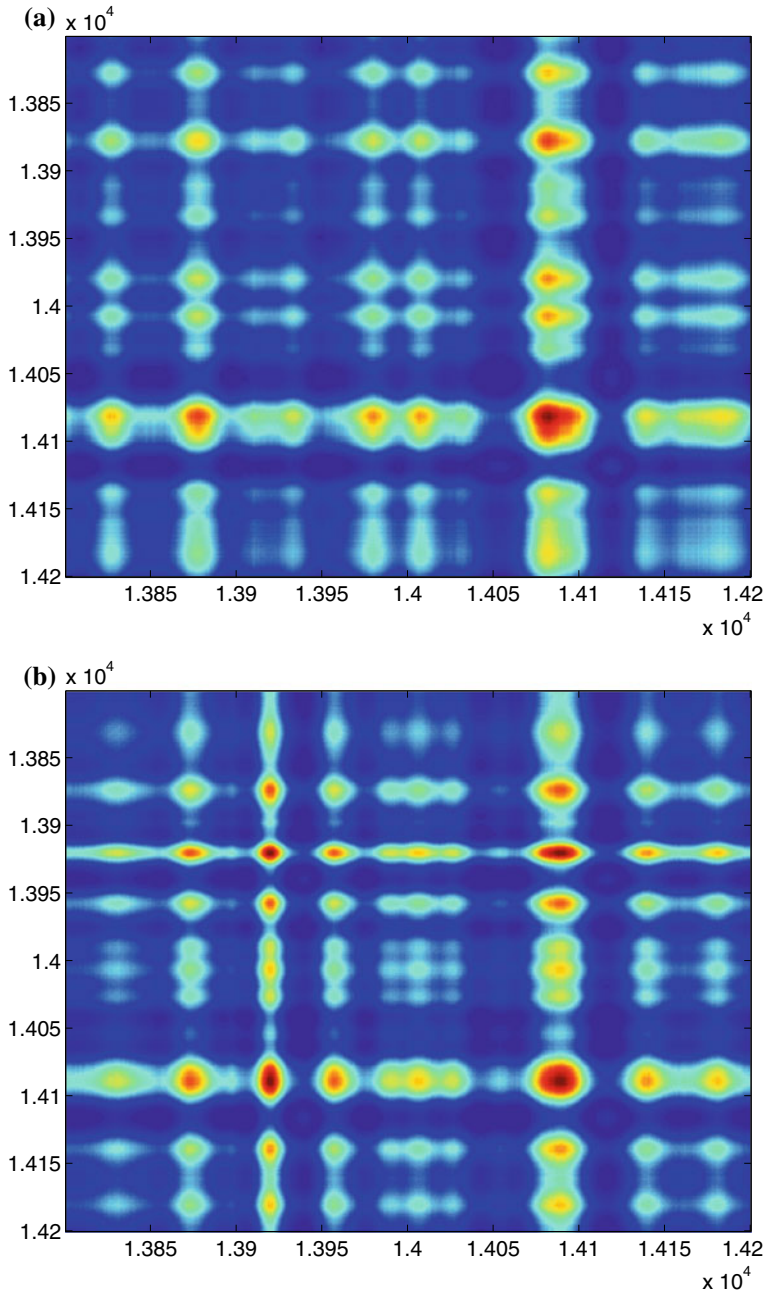


Fig. 6 2-D image created via adding voltages with the appropriate time offset. **a** No target. **b** With target in the field

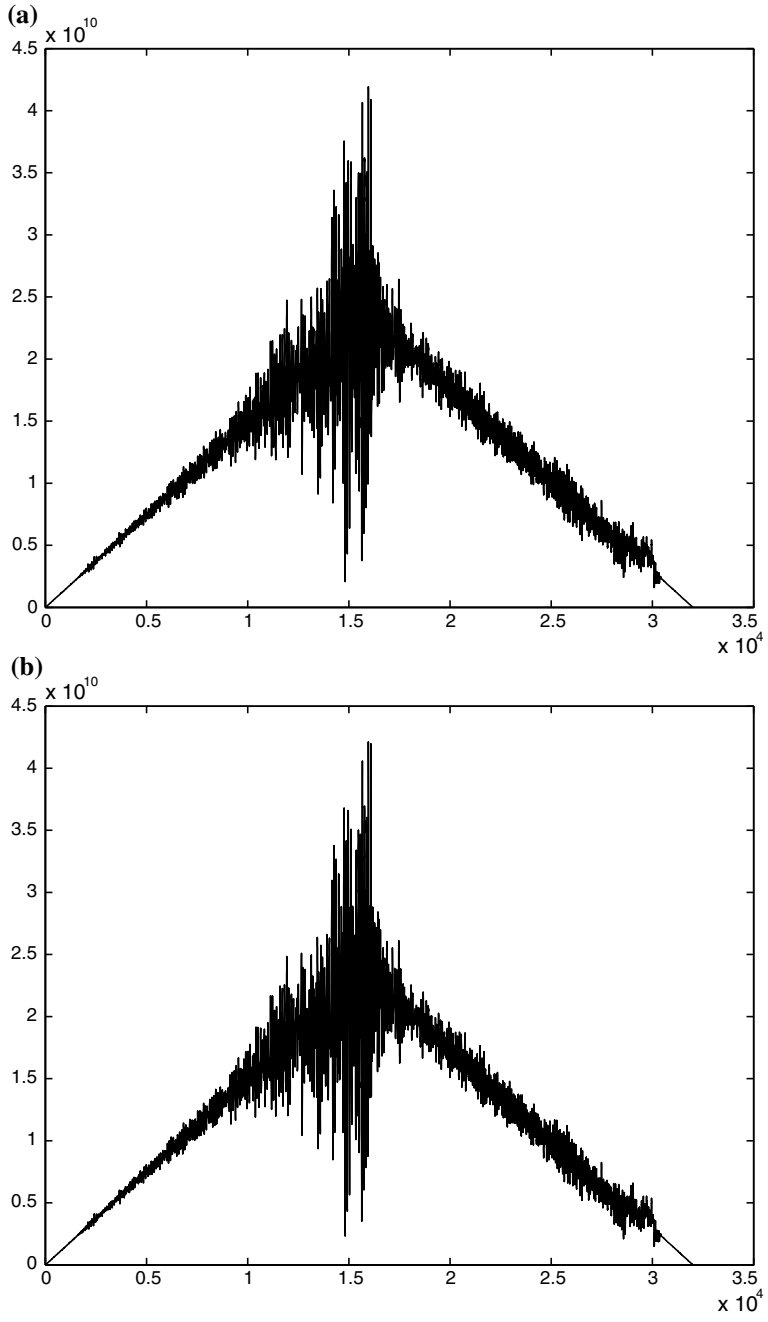


Fig. 7 The matched filter output **a** no target, and **b** with target

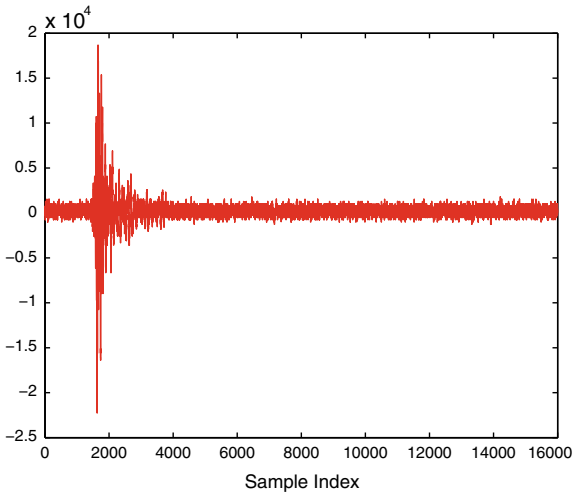


Fig. 8 UWB radar transmitted pulse (averaged over 100 pulses)

impulse response $h(t) = s^*(T_M - t)$ The time T_M at which the SNR is maximized is arbitrary; however, $T_M \geq t$ is required for $h(t)$ to be causal. Given the received echo $x(t)$ consisting of clutter, target, and noise components, the output $y(t)$ of the matched filter is given by the convolution between $x(t)$ and $h(t)$

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \tag{3}$$

$$= \int_{-\infty}^{\infty} x(\tau)s^*(\tau + T_M - t)d\tau \tag{4}$$

In this chapter, we choose $T_M = 16001$, and the matched filter outputs for received signal in Fig. 3a (without target) and signal in Fig. 3b (with target) are plotted in Fig. 7a, b respectively. Since the received echoes plotted in Fig. 3a, b are averaged over 100 pulses, the transmitted pulse $s(t)$ in (4) is obtained via averaging corresponding 100 transmission pulses and is plotted in Fig. 8. Observe Fig. 7a, b, it is impossible to perform target detection based on the matched filter output.

Why the matched filter approach does not work for UWB radar-based target detection? We further studied the UWB channel using CLEAN algorithm [6, 13, 46]. Based on the transmit pulse in Fig. 8 and received echo in Fig. 4a, we applied CLEAN algorithm and obtained the UWB channel (plotted in Fig. 9). Observe Fig. 9, the UWB channel has memory because it is a linear filter. However, the matched filter is derived based on the assumption that the radar channel has no memory. The memory in UWB radar channel causes inter-symbol interference of transmit pulse and makes the matched filter approach perform poor.

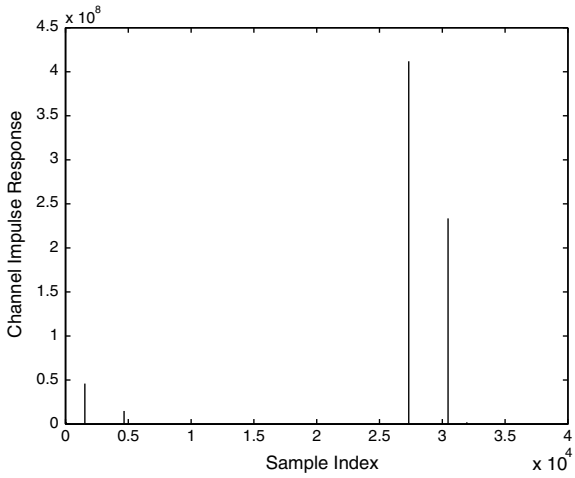


Fig. 9 The channel impulse responses for UWB channel using CLEAN method

4 Waveform Design and Diversity in Radar Sensor Networks

4.1 Coexistence of Radar Waveforms

In radar sensor networks (RSNs), radar sensors interfere with each other and the signal-to-interference ratio may be very low if the waveforms are not properly designed. In this chapter, we introduce orthogonality as one criterion for waveform design in RSN to make radars coexistence. In addition, since the radar channel is narrowband, we will also consider the bandwidth constraint.

In our radar sensor networks, we choose the CF pulse waveform, which can be defined as

$$x(t) = \sqrt{\frac{E}{T}} \exp(j2\pi\beta t) \quad -T/2 \le t \le T/2 \tag{5}$$

where β is the RF carrier frequency in radians per second. In radar, ambiguity function (AF) is an analytical tool for waveform design and analysis, which succinctly characterizes the behavior of a waveform paired with its matched filter. The ambiguity function is useful for examining resolution, side lobe behavior, and ambiguities in both range and Doppler for a given waveform [43]. For a single radar, the matched filter for waveform $x(t)$ is $x^*(-t)$, and the ambiguity function of CF pulse waveform is

$$\begin{aligned} A(\tau, F_D) &= \left| \int_{-T/2+\tau}^{T/2} x(t) \exp(j2\pi F_D s) x^*(t-\tau) dt \right| \\ &= \left| \frac{E \sin[\pi F_D (T - |\tau|)]}{T\pi F_D} \right| \quad -T \le \tau \le T \end{aligned} \tag{6}$$

We can simplify this AF in the following three special cases:

1. When $\tau = 0$,
$$A(0, F_D) = \left| \frac{E \sin(\pi F_D T)}{T \pi (F_D)} \right|; \tag{7}$$
2. when $F_D = 0$,
$$A(\tau, 0) = \left| \frac{E(T - |\tau|)}{T} \right|; \tag{8}$$
3. and when $\tau = F_D = 0$,
$$A(0, 0) = E \tag{9}$$

Note that the above ambiguity is for one radar only (no coexisting radar).

For radar sensor networks, the waveforms from different radars interfere with each other. We choose the waveform for radar i as

$$x_i(t) = \sqrt{\frac{E}{T}} \exp[j2\pi(\beta + \delta_i)t] \quad -T/2 \le t \le T/2 \tag{10}$$

which means that there is a frequency shift δ_i for radar i. To minimize the interference from one waveform to another, optimal values for δ_i should be determined to make the waveforms orthogonal to each other, i.e., let the cross-correlation between $x_i(t)$ and $x_n(t)$ be 0,

$$\int_{-T/2}^{T/2} x_i(t) x_n^*(t) dt = \frac{E}{T} \int_{-T/2}^{T/2} \exp[j2\pi(\beta + \delta_i)t] \exp[-j2\pi(\beta + \delta_n)t] dt \tag{11}$$
$$= E \operatorname{sinc}[\pi(\delta_i - \delta_n)T] \tag{12}$$

If we choose

$$\delta_i = \frac{i}{T} \tag{13}$$

where i is a dummy index, (12) can be written as

$$\int_{-T/2}^{T/2} x_i(t) x_n^*(t) dt = E \tag{14}$$

if $i = n$; or

$$\int_{-T/2}^{T/2} x_i(t) x_n^*(t) dt = 0 \tag{15}$$

if $i \neq n$.

Therefore, choosing $\delta_i = \frac{i}{T}$ in (10) yields orthogonal waveforms; i.e., the waveforms can coexist if the carrier spacing is a multiple of $1/T$ between two radar waveforms. In other words, orthogonality among carriers can be achieved by

separating the carriers by a multiple of the inverse of waveform pulse duration. With this design, all the orthogonal waveforms can work simultaneously. However, there may exist time delay and Doppler shift ambiguity which may interfere with other waveforms in RSN.

4.2 Interferences of Waveforms in Radar Sensor Networks

4.2.1 RSN with Two Radar Sensors

We are interested in analyzing the interference from one radar to another if there exist time delay and Doppler shift. For a simple case where there are two radar sensors (i and n), the ambiguity function of radar i (considering the interference from radar n) is

$$\begin{aligned} A_i(t_i, t_n, F_{D_i}, F_{D_n}) &= \left| \int_{-\infty}^{\infty} [x_i(t)\exp(j2\pi F_{D_i} t) + x_n(t-t_n)\exp(j2\pi F_{D_n} t)] x_i^*(t-t_i)dt \right| \\ &\le \left| \int_{-T/2+\max(t_i,t_n)}^{T/2+\min(t_i,t_n)} x_n(t-t_n)\exp(j2\pi F_{D_n} t) x_i^*(t-t_i)dt \right| \\ &\quad + \left| \int_{-T/2+t_i}^{T/2} x_i(t)\exp(j2\pi F_{D_i} t) x_i^*(t-t_i)dt \right| \qquad (16) \\ &= \left| \int_{-T/2+\max(t_i,t_n)}^{T/2+\min(t_i,t_n)} x_n(t-t_n)\exp(j2\pi F_{D_n} t) x_i^*(t-t_i)dt \right| \\ &\quad + \left| \frac{E\sin[\pi F_{D_i}(T-|t_i|)]}{T\pi F_{D_i}} \right| \qquad (17) \end{aligned}$$

To make the analysis easier, it is generally assumed that the radar sensor platform has access to the global positioning service (GPS) and the inertial navigation unit (INU) timing and navigation data. In this chapter, we assume that the radar sensors are synchronized and that $t_i = t_n = \tau$. Then, (17) can be simplified as

$$A_i(\tau, F_{D_i}, F_{D_n}) \approx \left| E\,\mathrm{sinc}[\pi(n-i+F_{D_n}T)] \right| + \left| \frac{E\sin[\pi F_{D_i}(T-|\tau|)]}{T\pi F_{D_i}} \right| \qquad (18)$$

We have the following three special cases:

1. If $F_{D_i} = F_{D_n} = 0$, and δ_i and δ_n follow (13), (18) becomes

$$A_i(\tau, 0, 0) \approx \left| \frac{E(T-|\tau|)]}{T} \right| \qquad (19)$$

2. If $\tau = 0$, (18) becomes

$$A_i(0, F_{D_i}, F_{D_n}) \approx \left|E\text{sinc}[\pi(n - i + F_{D_n}T)]\right| + \left|\frac{E\sin(\pi F_{D_i}T)}{T\pi F_{D_i}}\right| \tag{20}$$

3. If $F_{D_i} = F_{D_n} = 0$, $\tau = 0$, and δ_i and δ_n follow (13), (18) becomes

$$A_i(0, 0, 0) \approx E \tag{21}$$

4.2.2 RSN with M Radar Sensors

Our analysis on an RSN with two radar sensors can be extended to the case of M radars. Assuming that the time delay τ for each radar is the same, then the ambiguity function of radar 1 (considering interferences from all the other $M - 1$ radars with CF pulse waveforms) can be expressed as

$$A_1(\tau, F_{D_1}, \ldots, F_{D_M}) \approx \sum_{i=2}^{M} \left|E\text{sinc}[\pi(i - 1 + F_{D_i}T)]\right| + \left|\frac{E\sin[\pi F_{D_1}(T - |\tau|)]}{T\pi F_{D_1}}\right| \tag{22}$$

Similarly, we have the following three special cases:

1. $F_{D_1} = F_{D_2} = \cdots = F_{D_M} = 0$, and the frequency shift δ_i in (10) for each radar follows (13), then (22) becomes

$$A_1(\tau, 0, 0, \ldots, 0) \approx \left|\frac{E(T - |\tau|)]}{T}\right| \tag{23}$$

   Comparing it against (8), we notice that a radar may exist that can get the same signal strength as that of the single radar in a single radar system (no coexisting radar) when the Doppler shift is 0.
2. If $\tau = 0$, then (22) becomes

$$A_1(0, F_{D_1}, F_{D_2}, \ldots, F_{D_M}) \approx \sum_{i=2}^{M} \left|E\text{sinc}[\pi(i - 1 + F_{D_i}T)]\right| + \left|\frac{E\sin(\pi F_{D_1}T)}{T\pi F_{D_1}}\right| \tag{24}$$

   Comparing to (7), a radar in RSN has higher interferences when unknown Doppler shifts exist.
3. $F_{D_1} = F_{D_2} = \cdots = F_{D_M} = 0$, $\tau = 0$, and δ_i in (10) follows (13), then (22) becomes

$$A_1(0, 0, 0, \ldots, 0) \approx E \tag{25}$$

4.3 Radar Sensor Network for Collaborative Automatic Target Recognition

In RSN with M radars, the received signal for clusterhead (assume it is radar 1) is

$$r_1(u,t) = \sum_{i=1}^{M} \alpha(u) x_i(t - t_i) \exp(j2\pi F_{D_i} t) + n(u,t) \tag{26}$$

where $\alpha(u)$ stands for radar cross section (RCS), which can be modeled using nonzero constants for non-fluctuating targets and four Swerling target models for fluctuating targets [43]; F_{D_i} is the Doppler shift of the target relative to waveform i; t_i is the delay of waveform i, and $n(u,t)$ is the additive white Gaussian noise (AWGN). In this chapter, we propose a RAKE structure for waveform diversity combining, as illustrated by Fig. 10. The RAKE structure is so named because it reminds the function of a garden rake, each branch collecting echo energy similarly to how tines on a rake collect leaves. This figure summarizes how the clusterhead works. The received signal $r_1(u,t)$ consists of echoes triggered by the waveforms from each radar sensor, and $x_i^*(t - t_i)$ is used to retrieve the amplified waveform from radar i (amplified by the target RCS) based on the orthogonal property presented in Sects. 4.1 and 4.2, and then this information is time-averaged for diversity combining.

According to this structure, the received $r_1(u,t)$ is processed by a bank of matched filters, then the output of branch 1 (after integration) is

$$\begin{aligned} & Z_1(u; t_1, \ldots, t_M, F_{D_1}, \ldots, F_{D_M}) \\ &= \int_{-T/2}^{T/2} r_1(u,t) x_1^*(t - t_1) ds \qquad (27) \\ &= \int_{-T/2}^{T/2} [\sum_{i=1}^{M} \alpha_i(u) x_i(t - t_i) \exp(j2\pi F_{D_i} t) + n(u,t)] x_1^*(t - t_1) dt \qquad (28) \end{aligned}$$

Assuming $t_1 = t_2 = \cdots = t_M = \tau$, then based on (22),

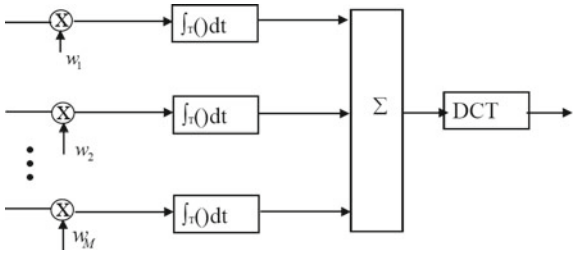


Fig. 10 Echo combining by clusterhead in RSN

$$Z_1(u;\tau,F_{D_1},\ldots,F_{D_M}) \approx \sum_{i=2}^{M}\alpha(u)E\,\mathrm{sinc}[\pi(i-1+F_{D_i}T)] + \frac{\alpha(u)E\sin[\pi F_{D_1}(T-|\tau|)]}{T\pi F_{D_1}} + n(u,\tau) \tag{29}$$

Similarly, we can get the output for any branch m ($m = 1, 2, \ldots, M$),

$$Z_m(u;\tau,F_{D_1},\ldots,F_{D_M}) \approx \sum_{i=1,i\neq m}^{M}\alpha(u)E\,\mathrm{sinc}[\pi(i-m+F_{D_i}T)] + \frac{\alpha(u)E\sin[\pi F_{D_m}(T-|\tau|)]}{T\pi F_{D_m}} + n(u,\tau) \tag{30}$$

Therefore $Z_m(u;\tau,F_{D_1},\ldots,F_{D_M})$ consists of three parts, namely signal (reflected signal from radar m waveform): $\frac{\alpha(u)E\sin[\pi F_{D_m}(T-|\tau|)]}{T\pi F_{D_m}}$, interferences from other waveforms: $\sum_{i=1,i\neq m}^{M}\alpha(u)E\,\mathrm{sinc}[\pi(i-m+F_{D_i}T)]$, and noise: $n(u,\tau)$.

We can also have the following three special cases for $|Z_m(u;\tau,F_{D_1},\ldots,F_{D_M})|$:

1. When $F_{D_1} = \cdots = F_{D_M} = 0$,

$$Z_m(u;\tau,0,0,\ldots,0) \approx \frac{E\alpha(u)(T-|\tau|)]}{T} + n(u,\tau) \tag{31}$$

   which means that if there is no Doppler mismatch, there is no interference from other waveforms.
2. If $\tau = 0$, (30) becomes

$$Z_m(u;0,F_{D_1},\ldots,F_{D_M}) \approx \sum_{i=1,i\neq m}^{M}\alpha(u)E\,\mathrm{sinc}[\pi(i-m+F_{D_i}T)] + \frac{\alpha(u)E\sin[\pi F_{D_m}T]}{T\pi F_{D_m}} + n(u) \tag{32}$$

3. If $\tau = 0$, and $F_{D_1} = \cdots = F_{D_M} = 0$, (30) becomes

$$Z_m(u;0,0,0,\ldots,0) \approx E\alpha(u) + n(u) \tag{33}$$

   Doppler mismatch happens quite often in target search where target velocity is not yet known. However, in target recognition, generally high-resolution measurements of targets in range ($\tau = 0$) and Doppler are available; therefore, (33) will be used for CATR.

How to combine all the Z_m's ($m = 1, 2, \ldots, M$) is very similar to the diversity combining in wireless communications to combat channel fading, and the combination schemes may be different for different applications. In this chapter, we are interested in applying the RSN waveform diversity to CATR, e.g., recognition that

the echo on a radar display is that of an aircraft, ship, motor vehicle, bird, person, rain, chaff, clear-air turbulence, land clutter, sea clutter, bare mountains, forested areas, meteors, aurora, ionized media, or other natural phenomena via collaborations among different radars. Early radars were "blob" detectors in that they detected the presence of a target and gave its location in range and angle, and radar began to be more than a blob detector and could provide recognition of one type of target from another [45]. It is known that small changes in the aspect angle of complex (multiple scatter) targets can cause major changes in the radar cross section (RCS). This has been considered in the past as a means of target recognition and is called *fluctuation of radar cross section with aspect angle*, but it has not had much success [45]. In [44], a parametric filtering approach was proposed for target detection using airborne radar. In [30], knowledge-based sensor networks were applied to threat assessment.

5 Sense-Through-Foliage Target Detection with Poor Signal Quality: A Sensor Network and DCT-Based Approach

As mentioned in Sect. 2, when the Barth pulse source was operated at low amplitude and the sample values are not obtained based on sufficient pulse response averaging (averaged over 35 pulses for each collection), significant pulse-to-pulse variability was noted and the return signal quality is poor. In Fig. 11a, b, we plot two collections with poor signal quality. Figure 11a has no target on range, and Fig. 11b has target at samples around 13,900. We plot the echo differences between Fig. 11a, b in Fig. 11c. However, it is impossible to identify whether there is any target and where there is target based on Fig. 11c. We observed the DCT-based approach failed to detect target based on one collection. Since significant pulse-to-pulse variability exists in the echoes, this motivate us to explore the spatial and time diversity using radar sensor networks (RSN).

In RSN, the radar sensors are networked together in an ad hoc fashion. They do not rely on a preexisting fixed infrastructure, such as a wireline backbone network or a base station. They are self-organizing entities that are deployed on demand in support of various events surveillance, battlefield, disaster relief, search, and rescue, etc. Scalability concern suggests a hierarchical organization of radar sensor networks with the lowest level in the hierarchy being a cluster. As argued in [14, 15, 33, 42], in addition to helping with scalability and robustness, aggregating sensor nodes into clusters has additional benefits:

1. conserving radio resources such as bandwidth;
2. promoting spatial code reuse and frequency reuse;
3. simplifying the topology, e.g., when a mobile radar changes its location, it is sufficient for only the nodes in attended clusters to update their topology information;
4. reducing the generation and propagation of routing information; and,
5. concealing the details of global network topology from individual nodes.

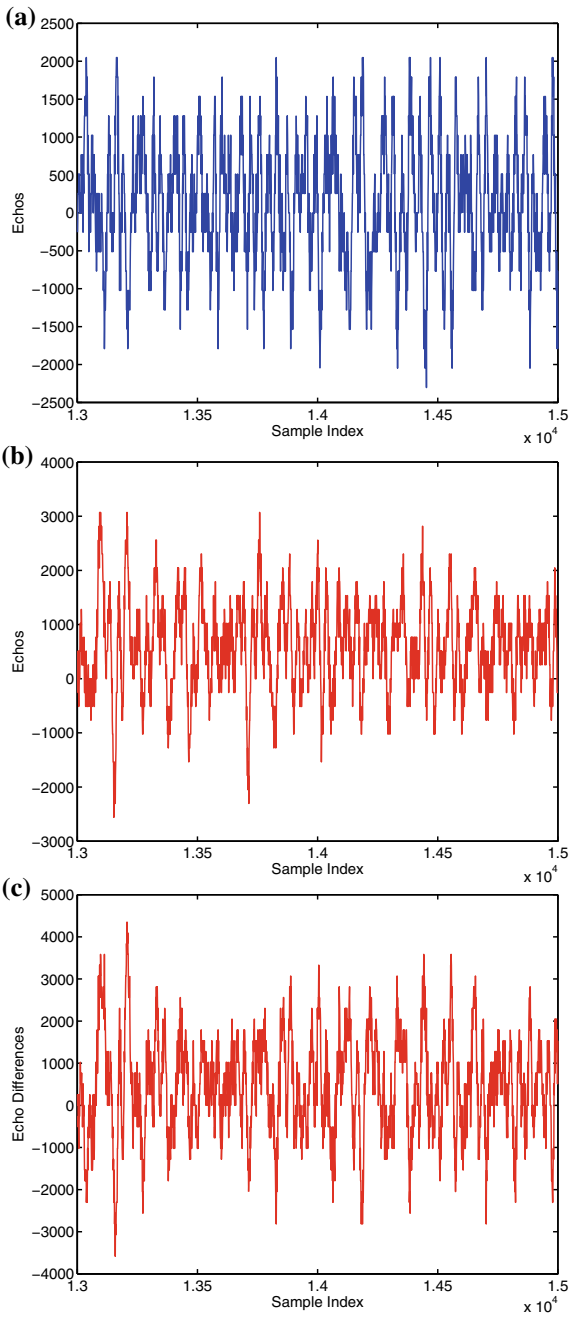


Fig. 11 Measurement with poor signal quality and 35 pulses average. **a** Expanded view of traces (no target) from sample 13,001 to 15,000. **b** Expanded view of traces (with target) from sample 13,001 to 15,000. **c** The differences between (**a**) and (**b**)

In RSN, each radar can provide their pulse parameters such as timing to their clusterhead radar, and the clusterhead radar can combine the echoes (RF returns) from the target and clutter. In this chapter, we propose a RAKE structure for combining echoes, as illustrated by Fig. 10. The RAKE structure is so named because it reminds the function of a garden rake, each finger collecting echo signals similarly to how tines on a rake collect leaves. The integration means time average for a sample duration T and it's for general case when the echoes are not in discrete values. It is quite often assumed that the radar sensor platform will have access to global positioning service (GPS) and inertial navigation unit (INU) timing and navigation data. In this chapter, we assume the radar sensors are synchronized in RSN. In Fig. 10, the echo, i.e., RF response by the pulse of each cluster-member sensor, will be combined by

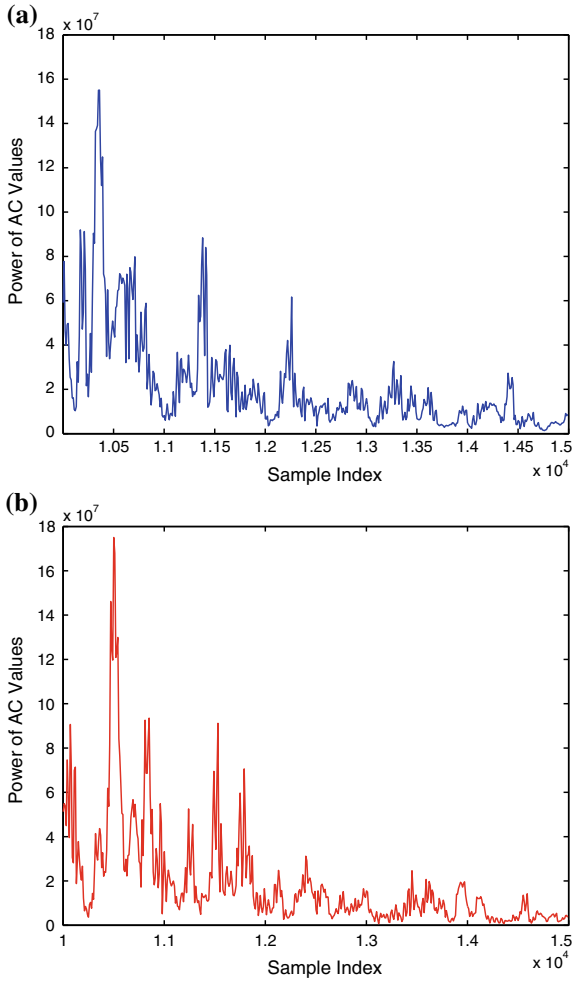


Fig. 12 Power of AC values based on UWB radar sensor networks and DCT-based approach. **a** No target. **b** With target in the field

the clusterhead using a weighted average, and the weight w_i is determined by the power of each echo $x_i(n)$ (n is the sample index),

$$w_i = \frac{E_i}{\sum_{i=1}^{M} E_i} \tag{34}$$

and

$$E_i = \text{var}(x_i(n)) + [\text{mean}(x_i(n))]^2 \tag{35}$$

We ran simulations for $M = 30$, and plot the power of AC values in Fig. 12a, b for the two cases (with target and without target), respectively. Observe that in Fig. 5b, the power of AC values (around sample 13,900) where the target is located is non-fluctuating (monotonically increase then decrease). Although some other samples also have very high AC power values, it is very clear that they are quite fluctuating and the power of AC values behaves like random noise because generally the clutter has Gaussian distribution in the frequency domain.

6 Human-Inspired Sense-Through-Foliage Target Detection

6.1 Human Information Integration Mechanisms

One of the great mysteries of the brain is cognitive control. How can the interactions between millions of neurons result in behavior that is coordinated and appears willful and voluntary? There is consensus that it depends on the prefrontal cortex (PFC) [36, 38]. A schematic diagram of some of the extrinsic and intrinsic connections of the PFC is depicted in Fig. 13 [36]. Many PFC areas receive converging inputs from at least two sensory modalities [7, 19]. For example, the dorsolateral (DL) (areas 8, 9, and 46) and ventrolateral (12 and 45) PFC both receive projections from visual, auditory, and somatosensory cortex. Furthermore, the PFC is connected with other cortical regions that are themselves sites of multimodal convergence. Many PFC areas (9, 12, 46, and 45) receive inputs from the rostral superior temporal sulcus, which has neurons with bimodal or trimodal (visual, auditory, and somatosensory) responses [5, 40]. The arcuate sulcus region (areas 8 and 45) and area 12 seem to be particularly multimodal. They contain zones that receive overlapping inputs from three sensory modalities [40]. Observe, for example, that mid-dorsal area 9 directly processes and integrates visual, auditory, and multimodal information. Regarding the functional model/mechanisms of different PFC areas (in Fig. 13): mid-dorsal area 9, dorsolateral area 46, and ventrolateral areas 12, 45, and orbital and medial areas 10, 11, 13, 14, different models and rules have been reported in the literature [10–12, 41].

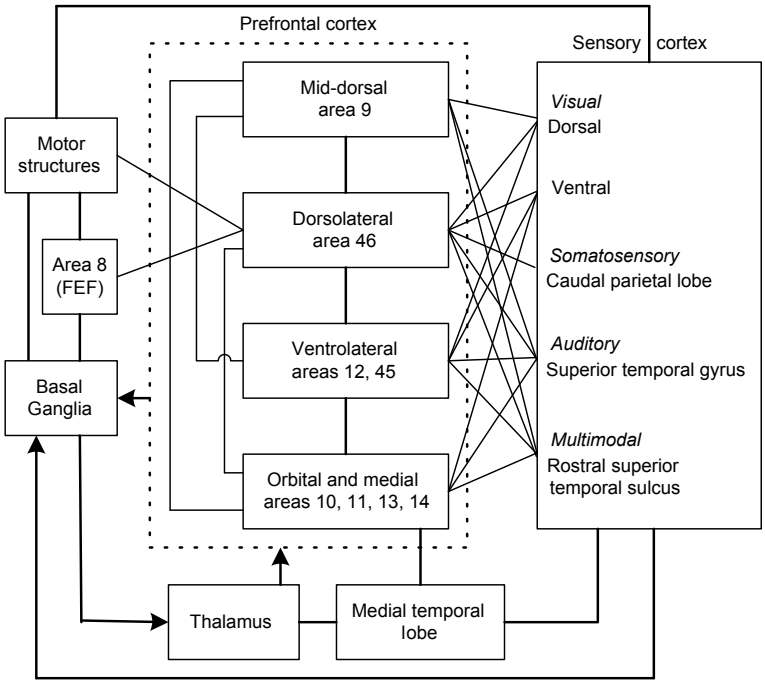


Fig. 13 Schematic diagram of some of the extrinsic and intrinsic connections of the PFC. Most connections are reciprocal; the exceptions are indicated by arrows. The frontal eye field (FEF) has variously been considered either adjacent to, or part of, the PFC

Recently, a maximum-likelihood estimation (MLE) approach was proposed for multi-sensory data fusion in human [12]. In the MLE approach [12], sensory estimates of an environmental property can be represented by $\hat{S}_j = f_i(S)$ where S is the physical property being estimated, f is the operation the nervous system performs to derive the estimate, and $\hat{S}$ is the perceptual estimate. Sensory estimates are subject to two types of error: random measurement error and bias. Thus, estimates of the same object property from different cues usually differ. To reconcile the discrepancy, the nervous system must either combine estimates or choose one, thereby ignoring the other cues. Assuming that each single-cue estimate is unbiased but corrupted by independent Gaussian noise, the statistically optimal strategy for cue combination is a weighted average [12]

$$\hat{S}_c = \sum_{i=1}^{M} w_i \hat{S}_i \tag{36}$$

where $w_i = \frac{1/\sigma_i^2}{\sum_j 1/\sigma_j^2}$ and is the weight given to the ith single-cue estimate, σ_i^2 is that estimates variance, and M is the total number of cues. Combining estimates by this

MLE rule yields the least variable estimate of S and thus more precise estimates of object properties.

Besides, some other summation rules have been proposed in perception and cognition such as soft-max rule: $y = (\sum_{i=1}^{M} x_i^n)^{\frac{1}{n}}$ [11] where x_i denotes the input from an input source i, and M is the total number of sources. In this paper, we will apply MLE and soft-max human brain information integration mechanisms to cognitive radar sensor network information integration.

6.2 Human-Inspired Sense-Through-Foliage Target Detection

We applied the human-inspired MLE algorithm to combine the sensed echo collection from $M = 30$ UWB radars, and then the combined data are processed using discrete cosine transform (DCT) to obtain the AC values. Based on our experiences, echo with a target generally has high and nonfluctuating AC values and the AC values can be obtained using DCT. We plot the power of AC values in Fig. 14a, b using MLE and DCT algorithms for the two cases (with target and without target), respectively. Observe that in Fig. 14b, the power of AC values (around sample 13,900) where the target is located is non-fluctuating (somehow monotonically increase then decrease). Although some other samples also have very high AC power values, it is very clear that they are quite fluctuating and the power of AC values behaves like random noise because generally the clutter has Gaussian distribution in the frequency domain.

Similarly, we applied the soft-max algorithm ($n = 2$) to combine the sensed echo collection from $M = 30$ UWB radars, and then used DCT to obtain the AC values. We plot the power of AC values in Fig. 15a, b using soft-max and DCT algorithms for the two cases (with target and without target), respectively. Observe that in Fig. 15b, the power of AC values (around sample 13,900) where the target is located is non-fluctuating (somehow monotonically increase then decrease).

We made the above observations. However, in real-world application, automatic target detection is necessary to ensure that our algorithms could be performed in real time. In Sect. 7, we apply fuzzy logic systems to automatic target detection based on the power of AC values (obtained via MLE-DCT or soft-max-DCT).

7 Fuzzy Logic System for Automatic Target Detection

7.1 Overview of Fuzzy Logic Systems

When an input is applied to a fuzzy logic system (FLS), the inference engine computes the output set corresponding to each rule [35]. The defuzzifer then computes a crisp

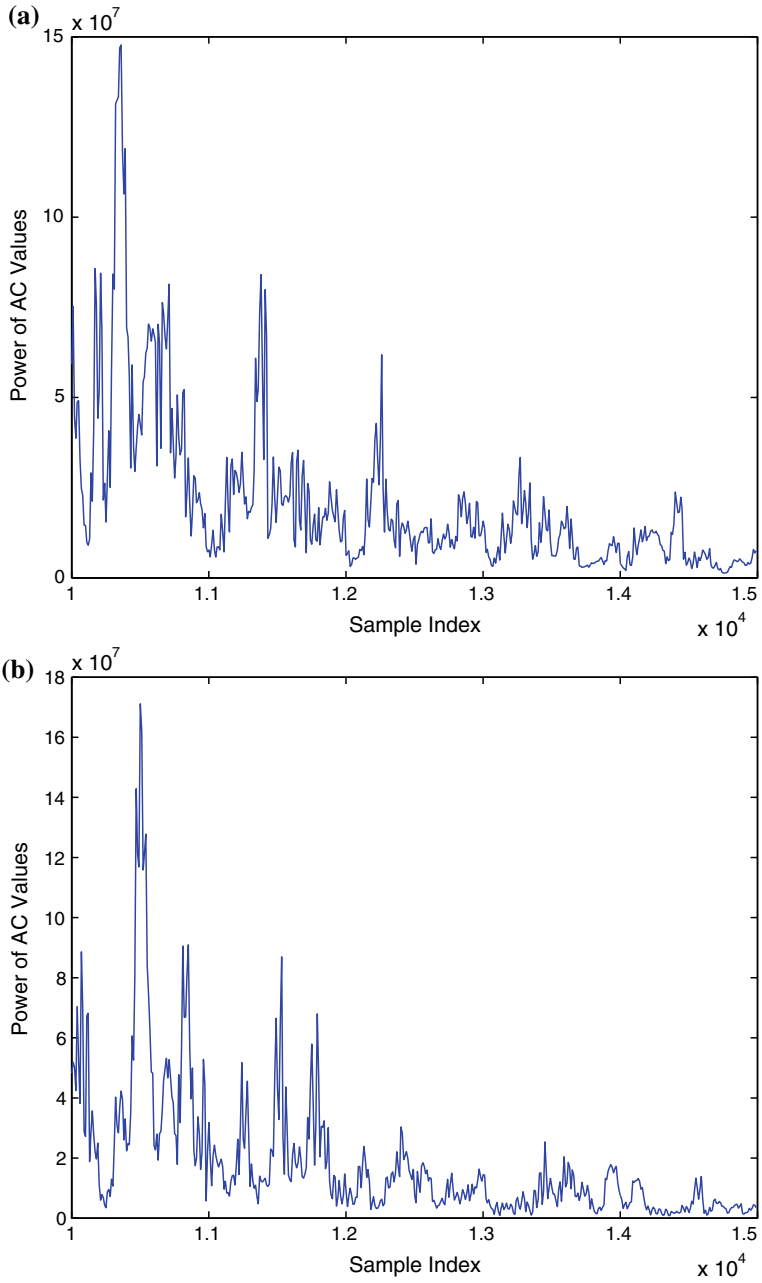


Fig. 14 Power of AC values using MLE-based information integration and DCT. **a** No target. **b** With target in the field

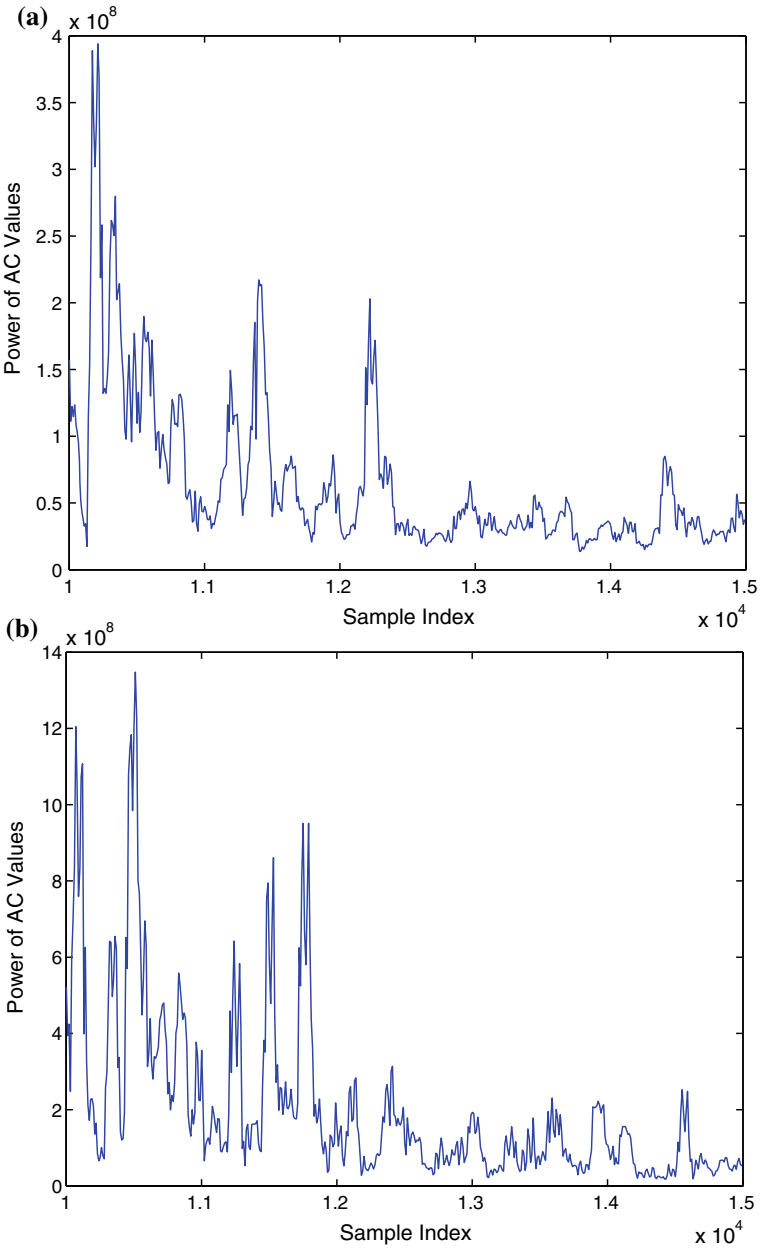


Fig. 15 Power of AC values using soft-max-based information integration and DCT. **a** No target. **b** With target in the field

output from these rule output sets. Consider a p-input 1-output FLS, using singleton fuzzification, *center-of-sets* defuzzification [35] and "IF-THEN" rules of the form

$$R^l : \text{IF } x_1 \text{ is } \mathrm{F}_1^l \text{ and } x_2 \text{ is } \mathrm{F}_2^l \text{ and } \ldots \text{ and } x_p \text{ is } \mathrm{F}_p^l, \text{ THEN } y \text{ is } \mathrm{G}^l.$$

Assuming singleton fuzzification, when an input $\mathbf{x}' = \{x_1', \ldots, x_p'\}$ is applied, the degree of firing corresponding to the lth rule is computed as

$$\mu_{\mathrm{F}_1^l}(x_1') \star \mu_{\mathrm{F}_2^l}(x_2') \star \cdots \star \mu_{\mathrm{F}_p^l}(x_p') = \mathscr{T}_{i=1}^{p} \mu_{\mathrm{F}_i^l}(x_i') \tag{37}$$

where $\star$ and $\mathscr{T}$ both indicate the chosen t-norm. There are many kinds of defuzzifiers. In this paper, we focus, for illustrative purposes, on the center-of-sets defuzzifier [35]. It computes a crisp output for the FLS by first computing the centroid, c_{G^l}, of every consequent set G^l, and, then computing a weighted average of these centroids. The weight corresponding to the lth rule consequent centroid is the degree of firing associated with the lth rule, $\mathscr{T}_{i=1}^{p} \mu_{\mathrm{F}_i^l}(x_i')$, so that

$$y_{cos}(\mathbf{x}') = \frac{\sum_{l=1}^{M} c_{\mathrm{G}^l} \mathscr{T}_{i=1}^{p} \mu_{\mathrm{F}_i^l}(x_i')}{\sum_{l=1}^{M} \mathscr{T}_{i=1}^{p} \mu_{\mathrm{F}_i^l}(x_i')} \tag{38}$$

where M is the number of rules in the FLS. In this paper, we design a FLS for automatic target recognition based on the AC values obtained using MLE-DCT or soft-max-DCT.

7.2 FLS for Automatic Target Detection

Observe that in Figs. 12 and 15, the power of AC values is quite fluctuating and has lots of uncertainties. FLS is well known to handle the uncertainties. For convenience in describing the FLS design for automatic target detection (ATD), we first give the definition of *footprint of uncertainty* of AC power values and *region of interest* in the footprint of uncertainty.

Definition 1 (*Footprint of Uncertainty*) Uncertainty in the AC power values and time index consists of a bounded region, that we call the *footprint of uncertainty* of AC power values. It is the union of all AC power values.

Definition 2 (*Region of Interest (RoI)*) An RoI in the footprint of uncertainty is a contour consisting of a large number (greater than 50) of AC power values where AC power values increase then decrease.

Definition 3 (*Fluctuating Point in RoI*) $P(i)$ is called a *fluctuating point* in the RoI if $P(i-1)$, $P(i)$, $P(i+1)$ are non-monotonically increasing or decreasing.

Table 1 The rules for target detection. Antecedent 1 is *centroid of a RoI*, Antecedent 2 is *the number of fluctuating points in the ROI*, and Consequent is *the possibility that there is a target at this RoI*

Rule #	Antecedent 1	Antecedent 2	Consequent
1	Low	Low	Medium
2	Low	Moderate	Weak
3	Low	High	Very weak
4	Moderate	Low	Strong
5	Moderate	Moderate	Medium
6	Moderate	High	Weak
7	High	Low	Very strong
8	High	Moderate	Strong
9	High	High	Medium

Our FLS for automatic target detection will classify each ROI (with target or no target) based on two antecedents: *the centroid of the ROI* and *the number of fluctuating points in the ROI*. The linguistic variables used to represent these two antecedents were divided into three levels: *low*, *moderate*, and *high*. The consequent—the possibility that there is a target at this RoI—was divided into five levels, *Very Strong, Strong, Medium, Weak, Very Weak*. We used trapezoidal membership functions (MFs) to represent *low*, *high*, *very strong*, and *very weak*; and triangle MFs to represent *moderate*, *strong*, *medium*, and *weak*. All inputs to the antecedents are normalized to 0–10.

Based on the fact the AC power value of target is nonfluctuating (somehow monotonically increase then decrease), and the AC power value of clutter behaves like random noise because generally the clutter has Gaussian distribution in the frequency domain, we design a fuzzy logic system using rules such as

$$R^l: \text{IF } \textit{centroid of a RoI } (x_1) \text{ is } \mathrm{F}_l^1, \text{ and } \textit{the number of fluctuating points in the ROI } (x_2) \text{ is } \mathrm{F}_l^2, \text{ then the possibility that there is a target at this RoI } (y) \text{ is } \mathrm{G}^l.$$

where $l = 1, \ldots, 9$. We summarize all the rules in Table 1. For every input (x_1, x_2), the output is computed using

$$y(x_1, x_2) = \frac{\sum_{l=1}^{9} \mu_{\mathrm{F}_l^1}(x_1)\mu_{\mathrm{F}_l^2}(x_2)c_{avg}^l}{\sum_{l=1}^{9} \mu_{\mathrm{F}_l^1}(x_1)\mu_{\mathrm{F}_l^2}(x_2)} \tag{39}$$

We ran simulations to 1000 collections in the real-world sense-through-foliage experiment and found that our FLS performs very well in the automatic target detection based on the AC power values obtained from MLE-DCT or soft-max-DCT and achieve probability of detection $p_d = 100\%$ and false-alarm rate $p_{fa} = 0$.

8 Conclusions and Future Works

In this chapter, we proposed a DCT-based approach for sense-through-foliage target detection when the echo signal quality is good, and a sensor network and DCT-based approach when the echo signal quality is poor. A RAKE structure which can combine the echoes from different cluster members is proposed for clusterhead in the RSN. We compared our approach with ideal case when both echoes are available, i.e., echoes with target and without target. We also compared our approach against the scheme in which 2-D image was created via adding voltages with the appropriate time offset as well as the matched filter-based approach. We observed that the matched filter-based could not work well because the UWB channel has memory. Simulation results show that our DCT-based scheme works much better than the existing approach, and our RSN- and DCT-based approach can be used for target detection successfully while the ideal case fails to do it. Inspired by human's innate ability to process and integrate information from disparate, network-based sources, we applied human-inspired information integration mechanisms to target detection in cognitive radar sensor network. Humans' information integration mechanisms have been modelled using maximum-likelihood estimation (MLE) or soft-max approaches. In this paper, we applied these two algorithms to cognitive radar sensor networks target detection. Discrete cosine transform (DCT) was used to process the integrated data from MLE or soft-max. We applied fuzzy logic system (FLS) to automatic target detection based on the AC power values from DCT. Simulation results showed that our MLE-DCT-FLS and soft-max–DCT-FLS approaches performed very well in the radar sensor network target detection, whereas the existing 2-D construction algorithm could not work in this study. For future works, we will collect more data with different targets and perform automatic target recognition besides target detection.

Acknowledgements This work was supported in part by U.S. Office of Naval Research (ONR) under Grant N00014-13-1-0043 and N00014-17-1-2733. The author would like to thank Dr. Sherwood W. Samn in AFRL for providing the data.

References

1. Barton, D.K.: Radar System Analysis and Modeling. Artech House, Boston, MA (2006)
2. Baggenstoss, P.: Adaptive pulse length correction (APLECORR): a strategy for waveform optimization in ultrawideband active sonar. IEEE Trans. Ocean. Eng. **23**(1), 1–11 (1998)
3. Baum, C.E., et al.: The singularity expansion method and its application to target identification. Proc. IEEE **79**(10), Oct 1991
4. Bell, M.R.: Information theory and radar waveform design. IEEE Trans. Inf. Theory **39**(5), 1578–1597 (1993)
5. Bruce, C., Desimone, C.R., Gross, C.G.: Visual properties of neurons in a polysensory area in superior temporal sulcus of the macaque. J. Neurophysiol. **46**, 369–384 (1981)
6. Cramer, R.J.-M.: An evaluation of ultrawideband propagation channels. Ph.D. Dissertation, USC (2000)

7. Chavis, D.A., Pandya, D.N.: Further observations on cortico-frontal connections in the rhesus monkey. Brain Res. **117**, 369–386 (1976)
8. Dill, C.: Foliage penetration (Phase II) field test: narrowband versus wideband foliage penetration. In: Final Report of Contract Number F41624-03-D-7001/04, July 2005–Feb 2006
9. Deng, H.: Synthesis of binary sequences with good correlation and cross-correlation properties by simulated annealing. IEEE Trans. Aerosp. Electron. Syst. **32**(1), Jan 1996
10. Dosher, B.A., Sperling, G., Wurst, S.A.: Tradeoffs between stereopsis and proximity luminance covariance as determinants of perceived 3D structure. Vision Res. **26**(6), 973–990 (1986)
11. Graham, N.V.S.: Visual pattern analyzers, vol. 646, pp. xvi. Oxford University Press, New York, NY, US (1989)
12. Hillis, J.M., Ernst, M.O., Banks, M.S., Landy, M.S.: Combining sensory information: mandatory fusion within, but not between, senses. Science **298**(5598), 1627–1630 (2002)
13. Hogbom, J.A.: Aperture synthesis with a non-regular distribution of interferometer baseline. In: Atronomy and Astrophysics Supplement Ser., vol. 15 (1974)
14. Hou, T.-C., Tsai, T.-J.: An access-based clustering protocol for multihop wireless ad hoc networks. IEEE J. Sel. Areas Commun. **19**(7), 1201–1210 (2001)
15. Iwata, A., Chiang, C.C., Pei, G., Gerla, M., Chen, T.W.: Scalable routing strategies for ad hoc networks. IEEE J. Sel. Areas Commun. **17**, 1369–1379 (1999)
16. Fitzgerald, R.: Effects of range-Doppler coupling on chirp radar tracking accuracy. IEEE Trans. Aerosp. Electron. Syst. **10**, 528–532 (1974)
17. Kershaw, D., Evans, R.: Optimal waveform selection for tracking system. IEEE Trans. Inf. Theory **40**(5), 1536–1550 (1994)
18. Kapoor, R., et al.: UWB radar detection of targets in foliage using alpha-stable clutter models. IEEE Trans. Aerosp. Electron. Syst. **35**(3), 819–834 (1999)
19. Jones, E.G., Powell, T.P.S.: An anatomical study of converging sensory pathways within the cerebral cortex of the monkey. Brain **93**, 793–820 (1970)
20. Liang, Q.: Biologically-inspired target recognition in radar sensor networks. In: International Conference on Wireless Algorithms, Systems, and Applications. Boston, MA, Aug 2009
21. Liang, Q.: Biologically-inspired target recognition in radar sensor networks. EURASIP J. Wireless Commun. Netw. Paper ID: 523435 (2010)
22. Liang, Q., Mendel, J.M.: Design interval type-2 fuzzy logic systems using SVD-QR method: rule reduction. Int. J. Intel. Syst. **15**(10), 939–957, Oct 2000
23. Liang, Q., Samn, S., Cheng, X.: UWB radar sensor networks for sense-through-foliage target detection. In: IEEE International Conference on Communications. Beijing, China, May 2008
24. Liang, Q., et al.: Opportunistic sensing in wireless sensor networks: theory and applications. IEEE Trans. Comput. **63**(8), 2002–2010 (2014)
25. Liang, Q.: Situation understanding based on heterogeneous sensor networks and human-inspired favor weak fuzzy logic system. IEEE Syst. J. **5**(2), 156–163 (2011)
26. Liang, J., Liang, Q.: Design and Analysis of distributed radar sensor networks. IEEE Trans. Parallel Distrib. Process. **22**(11), 1926–1933 (2011)
27. Liang, Q.: Automatic target recognition using waveform diversity in radar sensor networks. Pattern Recognit. Lett. (Elsevier) **29**(2), 377–381 (2008)
28. Liang, Q.: Radar sensor wireless channel modeling in foliage environment: UWB versus narrowband. IEEE Sens. J. **11**(6), 1448–1457 (2011)
29. Liang, Q., et al.: TDoA for passive localization: underwater versus terrestrial environment. IEEE Trans. Parallel Distrib. Process. **24**(10), 2100–2108 (2013)
30. Liang, Q., Cheng, X.: KUPS: knowledge-based ubiquitous and persistent sensor networks for threat assessment. IEEE Trans. Aerosp. Electron. Syst. **44**(3), July 2008
31. Liang, Q., Cheng, X., Samn, S.: NEW: network-enabled electronic warfare for target recognition. IEEE Trans. Aerosp. Electron. Syst. **46**(2), 558–568 (2010)
32. Liang, S.: Sense-through-wall human detection based on UWB radar sensors. Signal Process. **126**, 117–124 (2016)
33. Lin, C.R., Gerla, M.: Adaptive clustering in mobile wireless networks. IEEE J. Select. Areas Commun. **16**, 1265–1275 (1997)

34. Maherin, I., Liang, Q.: Multi-step information fusion for target detection using UWB radar sensor network. IEEE Sensors J. **15**(10), 5927–5937 (2015)
35. Mendel, J.M.: Uncertain Rule-Based Fuzzy Logic Systems. Prentice-Hall, Upper Saddle River, NJ (2001)
36. Miller, E.K., Cohen, J.D.: An integrative theory of prefrontal cortex function. Annu. Rev. Neurosci. **24**, 167 (2001)
37. Niu, R., Willett, P., Bar-Shalom, Y.: Tracking consideration in selection of radar waveform for range and range-rate measurements. IEEE Trans. Aerosp. Electron. Syst. **38**(2) (2002)
38. OReilly, R.C.: Biologically based computational models of high-level cognition. Science, Oct 2006
39. Papandreou, A., Boudreaux-Bartels, G.F., Kay, S.M.: Detection and estimation of generalized chirps using time-frequency representation. In: Twenty-Eighth Asilomar Conference on Signals, Systems and Computers, vol. 1, pp. 50–54, Oct 1994
40. Pandya, D.N., Barnes, C.: Architecture and connections of the frontal lobe. In: Perecman E. (ed.), The Frontal Lobes Revisited, pp. 41–72. IRBN, New York (1987)
41. Pelli, D.G.: Uncertainty explains many aspects of visual contrast detection and discrimination. J. Opt. Soc. Am. **A**(2), 1508–1532 (1985)
42. Perkins, C.E.: Chapter 4, cluster-based networks. In: Perkins, C.E. (ed.), Ad Hoc Networking, pp. 75–138. Addison-Wesley (2001)
43. Richards, M.A.: Fundamentals of Radar Signal Processing. McGraw-Hill Companies, New York (2005)
44. Roman, J., Rangaswamy, M., Davis, D., Zhang, Q., Himed, B., Michels, J.: Parametric adaptive matched filter for airborne radar applications. IEEE Trans. Aerosp. Electron. Syst. **36**(2), 677–692 (2000)
45. Skolnik, M.I.: Introduction to Radar Systems, 3rd edn. McGraw Hill, New York (2001)
46. Scholtz, R.A., Win, M.Z., Cramer, J.M.: Evaluation of the characteristics of the ultra-wideband propagation channel. Proc. Antenna Propag. Symp. **2**, 626–630 (1998)
47. Sowelam, S., Tewfik, A.: Waveform selection in radar target classification. IEEE Trans. Inf. Theory **46**(3), 1014–1029 (2000)
48. Stankovic, L., Thayaparan, T., Dakovic, M.: Signal decomposition by using the S-method with application to the analysis of HF radar signals in sea-clutter. IEEE Trans. Signal Process. **54**(11), 4332–4342 (2006)
49. Sun, Y., Willett, P., Lynch, R.: Waveform fusion in sonar signal processing. IEEE Trans. Aerosp. Electron. Syst. **40**(2) (2004)
50. Taylor, J.D.: Ultra-wideband radar technology, CRC Press (2001)
51. Taylor, J.D.: Introduction to Ultra-Wideband Radar Systems, CRC Press (1995)
52. Withington, P., Fluhler, H., Nag, S.: Enhancing homeland security with advanced UWB sensors. IEEE Microw. Mag. Sept, 2003
53. Wu, N., Liang, Q.: Sparse nested cylindrical sensor networks for internet of mission critical things. IEEE Int. Things J. Sept 2017. http://ieeexplore.ieee.org/document/8003289/
54. Wu, N., Liang, Q.: Coprime interpolation and compressive sensing for future heterogeneous network towards 5G. IEEE Access. **5**(1), 22004–22012 (2017)
55. Wu, N., Fangqi, Z., Liang, Q.: Evaluating spatial resolution and channel capacity of sparse cylindrical arrays for massive MIMO. IEEE Access **5**(1), 23994–24003 (2017)
56. Xu, L., Liang, Q.: Zero correlation zone sequence pair sets for MIMO radar. IEEE Trans. Aerosp. Electron. Syst. **48**(3), 2100–2113, July 2012
57. Xu, L., Liang, Q.: Radar sensor network using a new triphase coded waveform: theory and application. In: IEEE International Conference on Communications. Dresden, Germany, June 2009
58. Xu, L., Liang, Q.: Optimized punctured ZCZ sequence-pair set: design, analysis and application to radar system. EURASIP J. Wireless Commun. Netw. Paper ID: 254837 (2010)
59. Yarovoy, A.G., et al.: UWB radar for human being detection. IEEE Aerosp. Electron. Syst. Mag. **23**(5), 36–40 (2008)

60. Zhao, F., Wei, L., Chen, H.: Optimal time allocation for wireless information and power transfer in wireless powered communication systems. IEEE Trans. Veh. Technol. **65**(3), 1830–1835 (2016)
61. Zhao, F., Nie, H., Chen, H.: Group buying spectrum auction algorithm for fractional frequency reuses cognitive cellular systems. Ad Hoc Netw. **58**, 239–246 (2017)
62. Zhao, F., Li, B., Chen, H., Lv, X.: Joint beamforming and power allocation for cognitive MIMO systems under imperfect CSI based on game theory. Wireless Pers. Commun. **73**(3), 679–694 (2013)
63. Zhao, F., Sun, X., Chen, H., Bie, R.: Outage performance of relay-assisted primary and secondary transmissions in cognitive relay networks. EURASIP J. Wireless Commun. Netw. **2014**(1), 60 (2014)
64. Zhao, F., Wang, W., Chen, H., Zhang, Q.: Interference alignment and game-theoretic power allocation in MIMO heterogeneous sensor networks communications. Signal Process. **126**, 173–179 (2016)

Mobile Target Tracking with Multiple Objectives in Wireless Sensor Networks

Md Zakirul Alam Bhuiyan, Gary M. Weiss, Tian Wang and Geyong Min

Abstract Tracking mobile targets in wireless sensor networks (WSNs) is of utmost importance in surveillance applications. As it is often the case in prior work that the accuracy of tracking heavily depends on high accuracy in localization or distance estimation, which is never perfect in practice. These bring a cumulative effect on tracking (e.g., target missing). Recovering from the effect and also frequent interactions between nodes and a central server make tracking operation slow and energy-inefficient. Inspired by these, we design a tracking scheme, called `t-Tracking`, to address the target tracking problem in WSNs considering multiple objectives: low capturing time (e.g., the tracking time required to get around a target in a defined distance), high energy efficiency, and high quality of tracking (QoT). We propose a set of fully distributed tracking algorithms, which answer the query whether a target remains in a "specific area" (called a "face" in localized geographic routing, defined in terms of radio connectivity or local interactions of nodes). When a target moves across a face, the nodes of the face that are close to its estimated movements compute the sequence of the target's movements and predict when the target moves to another face. The nodes answer queries from a mobile sink called the "tracker," which follows the target along with the sequence. `t-Tracking` has advantages over prior work as it reduces the dependency on requiring high accuracy in localization and the frequency of interactions. It also timely solves the target missing problem caused by node failures, obstacles, etc., making the tracking robust in a highly dynamic

M. Z. A. Bhuiyan (✉) · G. M. Weiss (✉)
Department of Computer and Information Sciences, Fordham University, New York, NY 10458, USA
e-mail: mbhuiyan3@fordham.edu

G. M. Weiss
e-mail: gaweiss@fordham.edu

T. Wang
Department of Computer Science and Technology, Huaqiao University, Xiamen 361021, China
e-mail: wangtian@hqu.edu.cn

G. Min
Department of Mathematics and Computer Science, University of Exeter, Exeter, UK
e-mail: g.min@exeter.ac.uk

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_12

environment. We validate its effectiveness considering the multiple objectives in extensive simulations and in a system implementation.

Keywords Wireless sensor networks · Mobile target tracking · Distributed tracking algorithms · Face routing techniques · Wakeup mechanism · Quality of tracking · Energy efficiency

1 Introduction

A sensor is a low-cost device that detects changes in the environment and records the changes. It is typically capable of sensing, computing, and communication. A large number of sensors can collaborate to form a wireless sensor network (WSN), which can be used to monitor large areas effectively. Sensor nodes in a WSN constitute a wireless ad-hoc network, with one or a few sink nodes as the collection point(s) and bridge(s) to the central server (called the sink). Every node in the network may create data periodically, on demand of the sink, or triggered by events of interest. At the same time, every node may forward data that it receives toward sink nodes, which are often multiple hops away [1–9].

WSNs are increasingly being envisioned for collecting data, for example physical or environmental properties, from a geographical areas of interest. The applications of WSNs can be found in diverse fields such as survivable military surveillance systems (e.g., battlefield surveillance), environmental protection (e.g., habitat monitoring), industrial monitoring (e.g., machine equipment monitoring), monitoring healthcare (e.g., telemonitoring of human physiological data), personnel monitoring, home automation, and so on. One of the most important areas where the advantages of WSNs can be exploited is tracking mobile targets. Methods of tracking mobile targets have been gaining more and more attention due to their importance in employing wireless sensor networks (WSNs) for surveillance applications [10–21]. Sensors deployed for tracking schemes are capable of deducing kinematic characteristics, such as position, velocity, and acceleration of single or multiple targets of interest.

1.1 Motivation

We particularly discuss some practical examples that we target is in this work. The American Border Patrol operation [22] have tested both sensor-mounted UAV (unmanned aerial vehicle) and generic vehicle in conjunction with the existing static WSN (or so-called “virtual fence”) along the American/Mexican borders. Under such a scenario, the vehicle (or a sink) is not only able to follow a target of interest, but also overcome a situation when a “connectivity or coverage hole” appears on the virtual fence due to ground sensor faults. The sink may also use a follow-up verification about the target location due to the false positives generated from the sensors.

In a battlefield reconnaissance, fast tracking is required to detect an intrusion into restricted areas or to focus on evicting an enemy from an overseas territory (i.e., the border or harbor patrolling [12, 23]). This has a top security priority in many developed countries. In an investigation scenario, it may be used to watch a susceptible person, regarding his movements and activities, and to follow the person in order to obtain some specific information. Another scenario can be road patrolling, which is carried out to inspect the roadways and adjacent properties to detect illegal activities that may adversely affect the surveillance in road environment [24].

In many practical scenarios similar to the above, the movements of a target are relevant only to local areas and for a short period of time. This implies that such a scenario requires fast tracking operation and also the high quality of tracking (QoT). This QoT can be used as the quality of service (QoS) in a tracking system [25]. On the one hand, if one wishes to achieve tracking a target by sensor nodes that are already organized into local groups (unlike dynamic clusters or trees) before the network starts tracking operation, the tracking will be energy-efficient because such tracking operation does not require a central server interaction in tracking. On the other hand, in order to track (or capture/reach the target) timely in a surveillance application, the systems should demand a low tracking time (or capturing time). Regarding these in prior work, we argue that there still remains concerns for energy-constrained WSNs to be addressed.

1.1.1 Communication Concerns

Sensors are deployed to sense and record the detection and localization of a target into the data logger, or to send to a central server (or a sink, which may be deployed outside of the tracking area). In many prior works [14, 15, 17, 26–29], sensors are so often required to interact with the sink and reply with queries about a target and with locations. The sink is responsible for analyzing the data and sending feedback to the sensors in each step of tracking. Although those works come up with many practical solutions that advance the research of target tracking in WSNs, there still exists issues to be tackled, e.g., frequent interactions between the nodes and the sink, delay due to congestion, a single point of failure, etc. More importantly, the network area is usually divided in regions, cells, grid, clusters, trees, etc., to track the target in a distributed manner [17, 30–33]. Such division obviously incurs extra frequency of interactions and energy consumption. Sensor nodes near the sinks consume more energy and die first such that the tracking task may become critical and slow (Table 1).

1.1.2 Localization Concerns

In those scenarios, when a mobile target is under investigation, naturally the target does not cooperate with the investigators. In such a case, the tracking tasks (detection, localization, and query) entirely rely on the interactions between the sensors and the

Table 1 Abbreviation description

Abbreviation	Description
RSSI	Received signal strength indication
TOA	Time of arrival
TDOA	Time difference of arrival
AOA	Angle of arrival
MPS	Multipath signature
LSS	Least-squares scaling
MDS	Multidimensional scaling

investigators. More precisely, this relies on the sensor nodes in the local area around the target. Prior work, in which tracking heavily relies on high accuracy of localization, may not provide high QoT or may need to compromise between QoT and energy efficiency of WSNs. Because accuracy in localization or distance estimation is never perfect in practice and the estimated locations of nodes may have errors. A WSN can achieve localization by interpreting metrics of the target information transmissions such as the RSSI, TOA, TDOA, AOA, MPS, LSS, MDS, or their combinations (please see [32, 34–48]). Physical interference can be high in outdoor environments. Although these metrics are often used in tracking, QoT is affected when operating environments are dynamic. The localization suffers from secular biases due to effects of shadowing or multi-path propagation, radio occlusions, and decalibration, as well as large unbiased errors due to measurement noise. The inaccuracy in the localization cannot be eliminated even with a plenty of observation data.

1.1.3 Robustness Concerns

Due to the localization inaccuracy, the event of target missing problem often occurs, resulting in a significant energy consumption (e.g., to relocate the missing target). The situation becomes more serious in the case of a WSN that is too dense or sparse, or environmental noise is high. In prior work, many times it is not possible to track the target in a energy-efficient and timely manner, if the event of target missing is caused by node failures, coverage or connectivity holes, or physical obstacles, etc.

Motivated by the above concerns, we design a tracking scheme, called `t-Tracking`, with the aim to optimize multiple objectives. We enable sensors located in a local area near a target moving path to detect the target (see Fig. 1) and compute and store data locally; they can even reply with queries locally. Target tracking is made more efficient by exploiting a mobile sink which can always be around the detecting sensors or a little distance away (single to multihop) [49, 50]. The objectives are to reduce the capturing time (a new metric that measures the total tracking time required to get around of a target within a certain distance), to enhance the energy efficiency of the WSN and to ensure the QoT.

1.2 Our Scheme: `t-Tracking`

An entity, e.g., a respective authority that intends to follow a target, is called a *tracker*, which can also be called a *mobile sink* because it traverses through the network (see Fig. 1 for an example). A tracker is assumed to be a single generic source such as a mobile user or a respective authority. A *target* can be any mobile entity such as an enemy vehicle or an intruder. Thus, two mobile nodes, "Target" and "Tracker," are implemented. A WSN composed of a set of static sensor nodes is deployed in a plane, where the target moves in dynamic patterns. Through graph planarization, the WSN is organized into non-overlapping areas (called "faces" as shown in Fig. 1), which is usually carried out in localized geographic routing (particularly, in face routing) [50–52]. Each face (e.g., F_2), comprising a number of nodes, corresponds to a local area of the WSN.

When the tracker intends to follow a target, it queries the WSN. The nodes in the WSN are periodically clock synchronized to be in an awake, active, or inactive state. Each node has the capability of sensing, computing, and communicating. When a node of a face receives a query request, it checks with its neighboring nodes whether or not it is the closest to the target; if it is, it is elected as a *monitor* and one of its neighbors is elected as a *backup* (see Fig. 1). The monitor then works at the request of the tracker and sends information about the monitor itself, the backup, and the target, while the target traverses through the face. In the case that the monitor has any problem due to any reason, the backup takes the role of the monitor. Target detection and localization is mainly performed by the cooperation between the monitor and the backup.

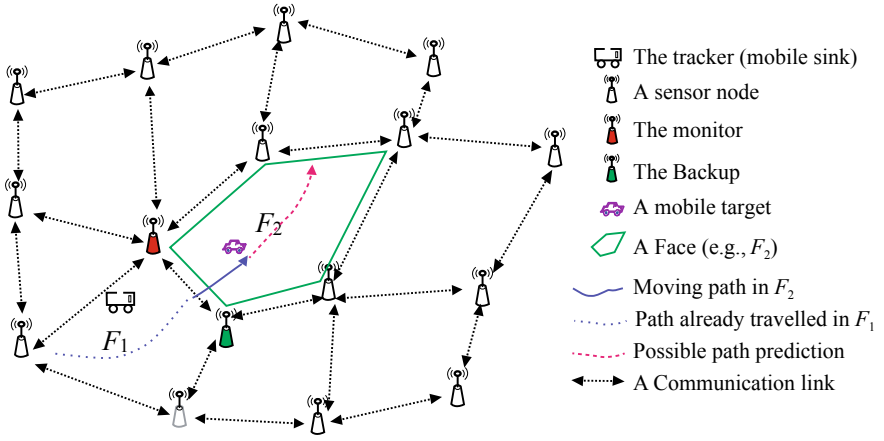


Fig. 1 An illustration of an intrusion into a restricted area, where an intruder (i.e., a vehicle) is pursued by an authority (i.e., a tracker), in order to observe its movements

The tracker then moves toward the monitor and queries for an update. If the target is still within the face, the monitor keeps tracking the target; at the same time, the monitor elects the next possible monitor and backup to be the new monitor and backup by using our prediction method. If the target has already moved out of the area of the face, the monitor informs the tracker about the new monitor and backup, and the tracker moves toward them. Monitor and backup are two common sensors of the current face and one of its adjacent faces. When the monitor finishes its task, it changes its state to the inactive state. This is also true for the backup. In this way, a special linked list of monitors, backups, and other nodes in a face is formed as time goes on. If both the monitor and the backup are viewed as one *logical node* at each time step of the tracking, this special linked list is simply a linear link of logical nodes.

1.3 Distinctive Advantages

Some distinctive advantages of `t-Tracking` help reduce the capturing time and energy consumption.

- We take the challenge to reduce the dependency on requiring high accuracy in localization.
- After estimating the target's detection, the monitor locally chronicles in its stack, and waits for the tracker request. The tracker does not need to wait for the detection information. The frequency of interactions between the nodes and tracker reduces.
- By employing a low complexity prediction of the next face, the number of nodes and faces required in tracking is minimized. The sensor's calculations are simplified and the interactions (communication) between the sensors (other than the monitor or backup) the tracker is minimized.
- The monitor does not transmit the target's information to interact with all the neighbors in faces except for the special events, e.g., target missing or failure event. Thus, there is no need of extra interactions between the nodes.
- The nodes in the faces, where there is no target, need not perform any interactions in tracking and can be inactive to save energy. After computation, the monitor transmits the target information to the tracker only when it receives a request.
  Another important objective of our work is to ensure the QoT. This tracking system is robust if there is an event of target missing problem. If there is the event of target missing, we allow neighbors that are the closest to the monitor in the current face to cooperate. Even in this case, if the target is not sensed, we allow all the neighbors in the face to relocate the target. If the target is still not sensed, we allow all the neighbors outside of the face to relocate the target. Even when all the neighbors fail, `t-Tracking` reverts to the initial detection state.

1.4 Contributions

To sum up, the original contributions of this chapter are as follows:

- To the best of our knowledge, we are the first to study the problem of target tracking under localized areas (faces) in a planarized WSN (see [53, 54]). We show how to organize the nodes into faces in order to track a target.
- We design `t-Tracking` to address the problem with multiple objectives, in which we employ a set of fully distributed tracking algorithms. In the algorithms, we put into practice two mobile nodes under the static WSN, in which the sink mobility is exploited.
- For a fast tracking operation, we use a simplified kinematics-based prediction to achieve "face prediction" where the QoT is guaranteed even when the localization is not so accurate. We design a sensor state transition model to save energy in each time step of tracking.
- We evaluate the performance of `t-Tracking` in extensive simulations and in a system setup with 20 Imote2 sensors. Results show that, `t-Tracking` reduces the capturing time from 52 to 91% (implying to have a quick operation in surveillance applications) and increase the energy saving of WSN by at least 4 times in comparison to prior work [13–15, 35], while ensuring the high QoT.

1.5 Organization

The rest of the chapter is structured as follows. Section 2 reviews the related work. Section 3 explains the problem setup and our objectives. Section 4 provides the tracking algorithms about the target's movement detection and face prediction. Section 5 discusses the tracking process and robustness. Section 6 offers the design of `t-Tracking`. Section 7 provides the performance analysis of `t-Tracking`. Performance analysis and evaluation are conducted in Sect. 8 and Sect. 9, respectively. Finally, Sect. 10 concludes this chapter and highlights some future work. We wish to note that we discuss some concerns/questions in this chapter that are essential to make more clear the target tracking through faces.

2 Related Work

Tracking moving target using WSN technology is a thought-provoking and well-established research area. Current advance has generated a large body of works that have addressed various aspects of the problem of target tracking [13–17, 26, 28, 31, 32, 34–36, 39, 55]. Although existing schemes advance the research of target tracking in WSNs with many prominent protocols and algorithms, there still exists issues to be addressed.

In centralized tracking schemes, sensors directly send reports about sensor information, target information, network maintenance (e.g., in case of sensor faults), or other associated information to a sink (which may be deployed outside of the tracking area). The sink usually analyzes the tracking information and produces meaningful outputs.

Considering constraints (e.g., energy, bandwidth) and real-time requirements in WSNs, distributed tracking systems are attractive. Prior distributed schemes improve difficulties usually found in the centralized tracking schemes [14, 15, 17, 26, 28, 30]. Real-time processing and collaboration between the nodes can be achieved, and the data transmission during tracking is reduced by the methods of the use of clusters or trees in tracking. In many of the existing works, dynamic clustering, which is triggered by an incoming target, is encouraged. On detecting a target, a node volunteers to be the leader and subsequently clustering occurs. Clustering, however, involves exchanging of information regarding the location of the nodes taking part in the cluster formation. Maintaining neighbors and gathering clustering information consumes time and energy. Forming clusters and assigning sensing tasks, after a target is detected, introduce a certain amount of time delay in tracking the target. However, there are also many distributed tracking algorithms in which the tracking operation is not fairly distributed, requiring the central interactions [10, 11, 26, 56].

Besides the methods of clusters and trees, the network area is also partitioned into regions, cells, grid, etc., to track a target in a distributed manner [17, 30–32, 37, 57]. There are huge interactions (the number of times each sensor talks to its neighboring nodes and the sink) during tracking operation. Coordinating different instructions and collecting data from the sensors in a hierarchical manner enable the sensors to interact with the neighboring nodes, the intermediate leaders, and the sink so often. Delay in target detection and tracking may be induced due to multihop communication and congestion in the network. All these result in a lot of energy consumption of the network. Moreover, sensor nodes near the sinks consume more energy and die first such that the tracking task may become critical and slow. When faults occur in tracking nodes, and connectivity/coverage holes or physical obstacles appear during tracking, mitigating them at once is more challenging. `t-Tracking` is a kind of scheme that is different from them, which organizes the nodes into non-overlapping areas (faces), achieved at the system initialization (not during the tracking). Each face has a small number of nodes. All the nodes over the WSN do not need to be further organized (unlike the clustering or grouping), before the target arrives to a specific face.

A lot of localization techniques exist, e.g., RSSI, ToA, TDoA, AoA, MPS, LSS, MDS, GPS, [35, 36, 40–46, 56]. Most prior work is based on the assumptions that the locations of sensor nodes are exact and data collection by sensor nodes is accurate. However, uncertainty is ubiquitous in WSNs due to factors such as the impreciseness of positioning systems, environmental noise, physical interference, limited sensitivity of sensing components, and so on. In fact, the localization or distance measurements from the nodes to a target are not accurate [32, 34–37, 43, 46], due to the above reasons in the practical sensing field. Among those, TOA measurements are easy to acquire, as each sensor only needs to identify a special

signal feature such as a known signal preamble to record its arrival time. Xu et al. considers the joint problem of mobile sensor navigation and mobile target tracking based on an improved TOA measurement model (labeled as iTOA) [35]. They include a general TOA measurement model that accounts for the measurement noise due to multipath propagation and sensing error. Based on the model, they propose a min-max approximation approach (MMA) to estimate the location and make prediction for tracking that can be efficient. In the situation that, the location or distance estimation has inaccuracy, QoT and energy efficiency is affected. Such inaccuracy may happen in every step of tracking.

He et al. [14, 15] designed and implemented a real-time target tracking framework, VigilNet, a large-scale outdoor WSN, which tracks, detects, and classifies targets in a timely and efficient manner. Through experiments, VigilNet shows that it can meet real-time requirements and provide fast tracking, and its trade-offs are validated. On the basis of the deadline partition method and the use of theoretical derivations to guarantee each sub-deadline, they make guided and engineering decisions to meet the end-to-end tracking deadline.

Some prediction-based methods [58–60] are used to predict the location of mobile targets and to allow a limited number of sensors to track a target, and the use of mobile agents for tracking. Typical examples of the prediction-based methods are, namely pheromones, Bayesian, and extended Kalman filter [26, 28, 30, 31, 35, 57, 61].

A closely related research effort [13], called "Forms," presents the use of a differential one-form and location service for target tracking and aggregate range queries, which is based on network planarization (particularly faces) in WSNs. Forms nicely discusses tracking through faces in terms of queries, local updates, and maintenance cost under target movement. The difference is in building a differential one-form on a planar graph. However, the proposed algorithms are mainly based on theoretical aspects, ignoring the communication and localization aspects in target tracking in WSNs. During each time of tracking step in Forms, a number of faces (that containing a large number of nodes in total) need to involve in tracking. Forms may be suitable for a very large-scale network (with more than thousands of nodes). It may not be suitable for a small- to medium-scale network. We found that if it is applied in a network with hundreds of nodes (that is deployed in rectangular field), most of the nodes need to involve in tracking operation. Achieving tracking information timely at the sink in case of tracking one or more target in a surveillance application can be difficult.

`t-Tracking` attempts to improve many of the above difficulties and shortcomings to a great extent. The idea of organizing "face-shaped" local areas of the WSN is previously used in routing strategy, but we utilize the idea in tracking application [53, 54]. To get the full advantages from this face-based tracking, we consider target tracking using face prediction. We found that it is non-trivial to determine exactly at which face a target is currently moving and then at which face the target may move. It requires low update and query cost (low interactions between the nodes), and a lower time in tracking. It is also useful for a small- to large-scale network.

3 Problem Setup and Objectives

In this chapter, we focus on *the problem of tracking a mobile target for surveillance applications through a planarized WSN*. We design `t-Tracking` to address the problem with multiple objectives. In this section, we first provide the preliminaries, including assumptions and term definitions. We then provide some models. Finally, we discuss our objectives.

3.1 Preliminaries

Suppose that a tracker and a target are moving in a two-dimensional planar space with a WSN covering the entire space. The sensor nodes are only capable of communicating with other nodes in its proximity. We assume that all the nodes have identical sensing range R_s. (For convenience's sake, the important notational conventions used in this chapter are summarized in Table 2.) For a node located at a point, a circle can be used that is centered at the point and has a radius R_s to represent the sensing circle of the node. The node can cover any point inside its sensing circle. However, this is a theoretical assumption that has little impact on the performance of a WSN in practice.

The WSN deployed in a convex region $\mathbb{A}$ is covered if any point in $\mathbb{A}$ is covered by at least one node u. Any two nodes u and v can communicate directly as long as their separation is not larger than R_c, where $|(u, v)|$ is the Euclidean distance between u and v. This is obtained by a network that is modeled as a graph $G = (V, E)$ by utilizing a well-known, distributed planarization algorithm, namely, a relative neighborhood graph (RNG) [6, 51, 52, 62, 63]. The graph $G(V, E)$ is the communication graph of a set of nodes V, where each node is represented by a vertex in V, and edge $(u, v) \in E$ if and only if $|(u, v)| \leq R_c$. We assume that $|(u, v)|$ may vary in a practical environment, since the radio signal strength of a WSN is highly dependent on the environment and can be irregular, due to various reasons, e.g., radio-opaque obstacles, transceiver differences, etc. In such a case, R_c can be set to the minimum $|(u, v)|$ and the boundary of its irregular communication region. Similar to this, we tackle difficulties found in this planarized network model.[1]

3.2 Exploration of Faces

By removing intersecting edges from G, we can obtain a connected planar subgraph $G' = (V, E')$ that maintains connectivity with fewer edges in RNG. G' has no intersecting edges. The equation form is as follows:

[1] Please see Appendix A for the detail about how we mitigate the difficulties.

Table 2 Notational conventions

Symbol	Description
t	A mobile target
T	A mobile tracker
$\mathbb{H}$	A whole period of tracking in a system run
H	A period of tracking ($H \in \mathbb{H}$)
h	A discrete time of a period ($h \in H$)
R_c	Communication range of a sensor node
R_s	Sensing range of a sensor node
l_i and L_i	ith location of t and T, respectively
l_i^s	ith sensor node location
$d(l_i^s, l_i)$	Euclidean distance between l_i^s and l_i
m	Euclidean distance between L_i and l_i
S_i	Observation (target's signal strength)
w_i	Noise energy
s_0	Active state (a sensor senses, computes, and communicates)
s_1	Awakening state (a sensor senses and computes)
s_2	Inactive state (a sensor's deep sleeping mode)
u, v, w	Nodes
v_i	ith sensor node, e.g., v_1, v_2
F_i	ith face, e.g., F_1, F_2
N	Total number of sensor nodes in a WSN
n	Total number of sensor nodes connected for t's detection
$S_{\max}$	Maximum number of nodes actively involve in tracking
T_p	A common toggle period for waking up
D_c	Staying awake (duty cycle)
E	Total energy consumption during target tracking
$E_{s,k}$	Total energy saving obtained by state transition
ΔE	Total energy saving during target tracking
v_T	Velocity of T, $v_T \in [0, v_T^{max}]$
v_t	Velocity of t, $v_t \in [0, v_t^{max}]$

$$\begin{aligned} &\forall u, v \in V : (u, v) \in E' \ \ iff \\ &\exists w \in V : \max\{d(u, w), d(v, w)\} < d(u, v) \end{aligned} \tag{1}$$

When a network graph has no crossing edges or links, and it is not unidirectional and disconnected, the graph is planar. In other words, planarization is equivalent to an edge removal process on G with connectivity preservation [64]. Such a graph without having links crossing each other can be achieved by using some cross-link detection and removal techniques [65].

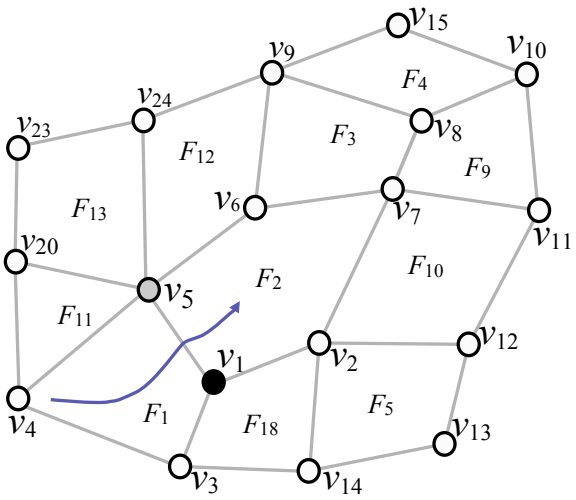


Fig. 2 An example of planarized network showing nodes and faces in the planar space

Face routing becomes an established technique to route packets using geographic information: Face routing guides packets along faces of the network communication graph and guarantees delivery when applied on a planar subgraph. These faces enclose void regions in the network on whose border one can find local minima. Figure 2 shows an example of a planarized network. Note that, as mentioned in Appendix A, the planarization in this work includes boundary node recognition techniques [64, 66] in order to reduce the difficulties in face generation. The figure shows planar faces (with reference to Fig. 1), in a planar neighborhood graph, pointing to tracking path lines between two nodes, indicating connectivity between them. Black- and gray-shaded nodes indicate the nodes in the active state.

The following terms and assumptions are in effect regarding in the system model in `t-Tracking`:

- The *tracker*, T, is assumed to be a single generic source such as a mobile user (e.g., a military personnel in battlefield), an investigator (can be in a vehicle or petrol car), or a fire-fighter, who is interested in obtaining information about a target. The tracker may show interests to the network and receive responses to the interests. The *target*, t, is a signal emitter, which is sensed and optionally observed by the WSN.
- The interest is a query about t. The query is implemented as one or more specific interests, e.g., requesting specific sensor node(s) (e.g., monitor and backup) to report a specific measurement as the request is received.
- All the nodes have the same sensing range and communication range in a regular case. Each node has a limited energy budget for sensing, computing, and making transmissions. But, in an irregular case, e.g., inaccuracy in neighbors' location information, the presence of obstacle, the nodes are assumed to be able to adjust R_c by choosing an upper power level.

- Upon detection, failed/disabled nodes are avoided. The system is able to avoid the loss of one or more nodes and to keep functioning as reliably as possible.
- *Capture point* denotes the point in tracking when T reaches the proximity of l_i^s (normally the monitor location) and $d(l_i^s, l_i)$ is less than a predefined value. In a real scenario, when T pursues t, T may reach t within very close range ($d(l_i^s, l_i) \leq \varepsilon$) and may capture t. ε is the predefined range, e.g., $\varepsilon \leq \frac{|R_s|}{2}$ or $\frac{|R_s|}{4}$. ε also can be said a "capture distance".

3.3 Models

3.3.1 Target Sensing Model

How to identify the target/object in the sensing coverage through faces is discussed in this subsection. We use a simple sensing model. Each node of the WSNs should find out t's location inside a face in $\mathbb{A}$, velocity, and direction. Suppose that $l_i \in \mathbb{A}$ is the location of the target at time $h \in [0, H]$. If there is t within R_s of a sensor node, the node senses the presence of t. It records the signal strength as follows:

$$S_i = \begin{cases} 0, & \text{if } r + r_e \leq d(s_i, l_i) \\ e^{-\lambda a^\beta} + w_i, & \text{if } r_e > |r - d(s_i, l_i)| \\ w_i, & \text{if } r - r_e \geq d(s_i, l_i) \end{cases} \tag{2}$$

Here, $r_e (r_e < r)$ is a measure of the uncertainty in t's detection, $a = d(l_i^s, l_i) - (r - r_e)$; λ and β are parameters that measure the detection when t is at a distance greater than r_e but is within a distance from the node. We normalize S_i, such that w_i has the standard Gaussian noise distribution, i.e., $w_i \sim (\mu, \sigma^2)$ where μ and σ^2 are the mean and variance of w_i, respectively [37]. Based on (2), ith sensor v_i located at l_i^s is able to detect t whenever $||l_i^s - l_i|| \leq R_s$. This detection may be unreliable, in the sense that failing to detect t does not imply that $||l_i^s - l_i|| \geq R_s$. Such false negatives, which can occur as a result of unmodeled occlusions in the operating environment, noise, or other factors, are assumed to be extremely common.

Given the target t's present location in $l_i(x_i, y_i)$ at h_i and t's previous location in $l_{i-1}(x_{i-1}, y_{i-1})$ at h_{i-1}, the Euclidean distance is denoted as follows:

$$d(l_i^s, l_i) = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \tag{3}$$

Since this work mainly focuses on tracking t, we ignore the details of the T's movement model.

3.3.2 Tracker Moving Model

Here, we discuss tracker T's movement model. t also moves in plane $\mathbb{A}$. We assume that T moves independently and in an optimal way. The movement of T is characterized by its speed, v_T. Suppose that at time h, the location of T is L_i. The speed of T is randomly chosen from a range $v_T \in [0, v_T^{max}]$ [12], where v_T^{max} is the maximum speed. T knows its location within $\mathbb{A}$. It has no sensors to directly detect t; it instead must rely solely on the communications from the WSN. The monitor and backup assist T to track t and to capture t at some time. T is assumed to have sufficient power and equipped with radios and processors, so that it can receive messages that are sent by the tracking nodes (monitor and backup in `t-Tracking`). It is capable to find the direction of a node location. T also uses the radio to transmit messages to inform the monitor v_i (and also the backup) of its presence. These messages are detected by the monitor v_i whenever t is in its R_s. Whenever T receives a message from the monitor located at l_i^s, it quickly moves to l_i^s and then moves from l_i^s to l_i. Based on these observations, we use the following strategy for T:

> *Moving with a maximum speed v_T^{max} along the shortest path in $\mathbb{A}$ from L_i to the close proximity to l_i^s and then to the close proximity to the l_i.*

Computing this motion takes computation time linear shortest path elements [67]. Our main interest in this work is in target tracking. Thus, we ignore the detailed design of the tracker. There is a number of schemes nicely describing the sink mobility models [12, 49, 50, 67–72] in terms of various concerns (e.g., the mobility pattern, routing techniques, data delivery to a mobile sink, latency in the delivery, connectivity, and delivery constraints.), which we follow when modeling the mobility of T in this work. Particularly, a scheme, *localized geographic routing to a mobile sink with guaranteed delivery*, greatly helps us in this work [50].

3.4 Objectives

To measure the performance of `t-Tracking`, three of the important metrics are as follows:

3.4.1 Decreasing the Target Capturing Time [C_T]

This is measured by the total tracking time that T requires to capture t at a certain location or a minimum distance. The reason for considering such a new metric is that it can indicate how fast a tracking system can detect and get a close proximity of t. Measuring such a metric can provide an insight into the tracking in border or road patrolling, military tracking applications, etc., where some time T is physically required to capture t. When calculating the capturing time, we take into account the target detection delay.

Whenever T is informed by the monitor v_i located at l_i^s about t's detection, T approaches to capture t. We need to mention that there is two-phase time duration that T spends in capturing t.

- This is the time in which T moves fast and passes a relatively short/long distance between itself and the proximity of l_i^s.
- This is the time in which T moves from the proximity of l_i^s to a close proximity of l_i.

As time increases, the relative emphasis placed on the first phase by m decreases. Thus, we consider the capturing time C at a capturing point, which measures the length of the first phase, and is defined by

$$C = \min\{h \in [0, H] \quad | \quad (m|d(l_i^s, l_i) \leq \varepsilon)\}, \tag{4}$$

The Euclidean distance between T and t, $m = ||L_i - l_i)||$, including the distance between L_i and l_i^s and $d(l_i^s - l_i))$. Hence, the total capturing time denoted by C_T is the total tracking time required by T from the tracking operation start to capture t at a capturing point in a system run, which is equal to the number of total tracking intervals spent plus C.

3.4.2 Increasing the Energy Efficiency of the WSN

This is measured by the total amount of energy saving denoted by ΔE. We seriously take into account the energy consumed by sensor nodes in every time step of tracking and sensor node duty cycle. We then estimate the total amount of energy saving during tracking. That is to mention, ΔE is attained by two-phase energy saving: (i) E_{s_k}—the amount of energy saving by each sensor low-power state transition ($s_k = s_0, s_1$, and s_2); (ii) E_{tr}—the amount of energy saving achieved during each time period ($h \in H$) of in tracking (e.g., in detection, localization, prediction) by reducing the frequency of interactions between nodes and between the nodes (monitor and backup) and T.

3.4.3 Ensuring the Quality of Tracking (QoT)

As the issue of the quality of service (QoS), QoT is measured by the rate of successful tracking steps (rSTS) in a system with robustness against all difficulties, including the presence of high localization errors, sensor faults, physical obstacles. We think that these difficulties remain to some degrees in tracking in the real operating environment.

Remarks The above three performance metrics, C_T, ΔE, and the QoT, are at least partially in conflict with one another. Intuitively, one can imagine that (i) a decrease in C_T or in the QoT results in a corresponding increase in ΔE, or (ii) a decrease in C_T or an increase in ΔE may result in a corresponding decrease in the QoT, i.e., the use of Pareto optimality concepts in tracking in a WSN. However, our simulation

and experiment results confirm that a decrease in C_T or an increase in ΔE generally results in a corresponding increase in the QoT. This tradeoff motivates us to treat the problem as a multiple-objective optimization problem.

4 Tracking Algorithms

In this section, we first show how we organize the nodes of a planarized WSN into faces for the purpose of tracking. Then, we present algorithms for the computation of target moving sequence through faces and use the sequence for face prediction. Finally, we provide energy-saving transition techniques.

4.1 Rules for Node Organization into Faces for Tracking

The planar subgraph consists of faces which are enclosed and bounded, in face-shaped regions [73], as described earlier. Each subgraph may have one or more faces. Exploration of a single face by any node v_i (such as the monitor) can be done in a localized way by applying the well-known *left-hand rule* that requires the message to traverse the edge. This edge lies the next in the counterclockwise direction from the previously visited edge. The *right-hand rule*, in contrast, requires the message to travel the edge lying next in clockwise direction. By utilizing the concept of faces as "local areas" from localized geographic routing (particularly, the face routing) techniques, we consider Fig. 2 as a representative example for the purpose of tracking in `t-Tracking`. We want to provide some definitions that are used throughout this chapter.

Definition 4.1 (*Adjacent Neighbors*) The neighboring nodes next to a specific node, which are directly connected, are called adjacent neighbors.

Definition 4.2 (*Face Neighbors*) All of the neighboring nodes inside a specific face F_i (where t is detected and is moving) of a specific node are called immediate neighbors.

Definition 4.3 (*Immediate Neighbors*) Those neighboring nodes, which are the adjacent neighbors (directly connected) and also the face neighbors inside face F_i of a specific node, are called immediate neighbors.

Definition 4.4 (*Distant Neighbors*) Those neighboring nodes, which are not the adjacent neighbors but the face neighbors inside face F_i of a specific node, are called distance neighbors.

Definition 4.5 (*Neighboring faces*) All those faces, which are corresponding to a specific node, are called neighboring faces. Sometimes, it may also include all those faces, which are the adjacent faces of a specific face F_i.

As T wishes, it issues a message to the WSN to assist to track t. Suppose that ith node v_i is any node of the WSN that can detect t, as t appears in the vicinity of v_i (e.g., v_1). Based on the detection probability (P_d), v_1 becomes the monitor. Then, v_1 starts finding face ith face F_i on which t appears. According to the WSN planarization at the system initialization, ith node v_i already has the information about its neighboring nodes and faces. From there, v_1 further updates the information about faces.

Let face F_i be the F_2. Then, faces, F_1, F_{12}, F_3, F_{10}, and F_{18} are called the *neighboring faces* of face F_2. Node v_1 of face F_2 corresponds to three adjacent faces, namely F_1, F_2, and F_{18}, as shown in Fig. 2. F_1 and F_{18} are called *neighboring faces* of v_1. The nodes in the three adjacent faces in G' are—(v_1, v_3, v_4, v_5), (v_1, v_5, v_6, v_7, v_2), and (v_1, v_2, v_3). v_1 has only three direct neighboring nodes v_2, v_3, and v_5, but here we only consider the neighboring nodes with respect to t's location in F_2. Thus, the nodes v_5, v_6, v_7, and v_2 in F_2 are called v_1's *face neighbors*. v_5 and v_2 in F_2 are called v_1's *immediate neighbors*. The rest of v_1's neighboring nodes, v_6 and v_7, are called *distant neighbors*. One (e.g., v_2) of the two immediate neighbors becomes the backup, as the combined detection probability (P_d^c) between the monitor and the neighboring node is the best.

Therefore, the monitor and backup (v_1 and v_2) are the active nodes of F_2 for a period of tracking and some of the distant neighbors may already detect t. The computation of t's moving sequence and tracking is mainly performed by the two nodes. They connect T. The distant neighbors can assist the monitor in tracking if needed, e.g., in the case of t's missing. The number of nodes in F_2 is five. The number of nodes in F_1 is four. In normal tracking, if t moves from F_1 to F_2, the number of nodes needed to involve in tracking is four to five, respectively. The two active nodes (monitor and backup) whose frequency of interactions can be higher than that of other nodes. Among them, the monitor usually has more interactions than the backup node. Thus, the number of total nodes required in H is minimized in `t-Tracking`. Similarly, node v_5 has five adjacent faces with eleven neighbors. If we considered all of the faces and nodes that correspond to a node (e.g., v_5), the frequency of interactions in the WSN would be high, resulting in a high energy consumption.

The above rules are extracted from the localized geographic routing techniques, by which all static nodes have locally accessible information about their neighbors without the use of exact locations [50, 74–76]. Such localized algorithms have been developed in the past for different contexts [52, 73, 74]. The main difference is that we do not consider the packet transmission from a node of a face to a destination node of another face (which is usually a long distance away). This implies that a packet does not travel several faces on the way to the destination node. We restrict it to a face (e.g., F_2 where t is moving) and neighboring faces only. This specifies that the nodes of faces around t are only organized. There is no need for any strong dependency on the localization, as this tracking application intrinsically finds the neighbors in the plane.

4.2 Target Detection and Target Moving Face Detection

In this subsection, we show how t can be tracked through the faces and present subsequent algorithms, including target detection, movement sequence, and face prediction.

4.2.1 Target Detection

In `t-Tracking`, each node is enabled to independently detect and process t's information locally, and collaborate with its neighbors in faces if the t happens within its R_s. Hence, t may be potentially detected by any node within their R_s. More precisely, any node which belongs to one or more faces of the WSN can detect t.

Algorithm 1: Target Detection

Input: A WSN of N sensor nodes observing the target t;
Output: t's location l_i at time h;
for each node v_i of the WSN at the s_1 state **do**
  Listen to the environment and start sensing;
  Measure S_i;
  if t is found **then**
    Change the status to s_0 state;
    Compute P_d;
    Run t's moving face detection algorithm;
    Compute current location l_i;
end for

Definition 4.6 (*Detection probability* (P_d)) P_d is the probability that a sensor node reports the presence of a t when t is within its R_s.

Each node estimates a detection probability P_d, which is *the probability that a node reports the presence of t when t is within its* R_s. It is estimated for the slow t to the fast t. The face strategy helps us to find t's moving since the monitor and backup share a part or the whole area of the face where the t is moving. Suppose that t enters the tracking area from point 0 to L meters. The area can be rectangular or semi-circular. Thus, the detection probability is stated as follows:

$$P_d = D_c + \frac{\pi r L}{(2L + \pi r/2) \cdot v \cdot T_p} \tag{5}$$

In the case of `t-Tracking`, we therefore need to guarantee that t is detected before it leaves the monitor's sensing range. We take $L = R$. Hence,

$$P_d = D_c + \frac{\pi r R}{(2R + \pi r/2) \cdot v \cdot T_p} \tag{6}$$

More details about the derivation of both slow and fast mobile targets can be found in [15].

Algorithm 1 gives the pseudocode of t's detection by using (2) and localize it at a location l_i based on P_d at the first time. Once t is localized, the face detection algorithm begins.

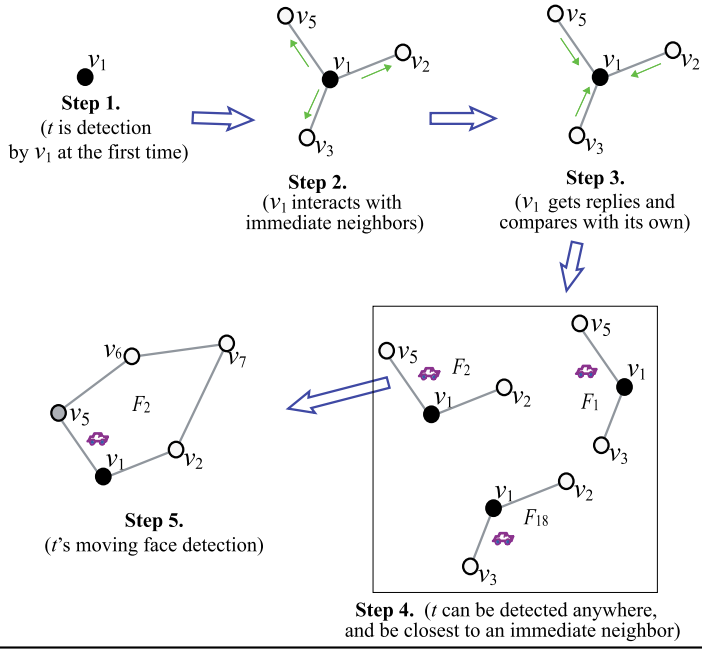
4.2.2 Target's Moving Face Detection

In this subsection, we discuss how a node decides on t in which specific face t is currently moving and then localize t inside the face at l_i for the first time. Algorithm 2 is presented in Fig. 3 for this purpose: There are several small steps at the beginning of the target tracking operation that are described as follows:

Step 1: There is t detected by node v_1 at h somewhere in the WSN by using P_d. Similarly, some of neighboring nodes (e.g., v_5, v_3, v_4, v_6, and so on) of v_1 might be able to detect t at h and have P_d in some extents.

Step 2: v_1 first interacts with its adjacent neighbors by issuing a request, containing the information that t is in the range, P_d, and $d(l_i^s, l_i)$. There are three adjacent neighbors of v_1: v_5, v_2, and v_3.

Algorithm 2: t's moving face detection at the first time (in which specific face t is currently moving.)



Step 3: After receiving the request messages from all of its neighbors (including the adjacent ones), node v_i compares its P_d with another node v_j that are paired up with it, e.g., $v_1 \leftrightarrow v_2$, $v_1 \leftrightarrow v_3$, $v_1 \leftrightarrow v_5$.

Step 4: Among all the neighbors, v_5 or v_2 has the second best detection probability that is the immediate neighbors. v_3 may have the lower detection probability than that of v_5 or v_2. Thus, t should be the inside F_2, instead of in F_{18} or in F_{11}. To know the pair of the nodes, which have the best detection probability, we consider to combine the detection probabilities, denoted by P_d^c, of each pair of the nodes. If one wishes, angle between the two or three adjacent neighboring nodes can be estimated to decide in which face t is in. Finally, we set three conditions for any pair of nodes (e.g., v_1 and v_5) to be the monitor and backup: (i) P_d^c should be higher than other pairs of nodes; (ii) they should be the adjacent and also be the immediate neighbors; (iii) they should be in the same face F_i (e.g., F_2).

Step 5: As the monitor and backup, v_1 and v_5 update the information of F_2 and neighboring faces by following the rules described earlier. Then, the complete face F_i can be detected and the nodes of the faces can be organized to track t.

Note that the above steps are used for face F_i detection at the first time. t's tracking can be easier in the WSN afterward, as another two nodes of F_2 (e.g., v_1 and v_5) compute t's movements and face prediction when t moves from F_i to F_j.

4.3 Computing Target Moving Sequence Through Faces

4.3.1 Target Movement Sequence Inside a Face

In a practical environment, t may move in complex and stochastic ways in any direction from face F_i to a future face F_j, as shown in Fig. 3a. t's velocity is unpredictable and it may be impossible to explicitly express the velocity. We enable the monitor and backup to compute the moving sequence based on possible locations of t according to its current motion state and tracks t when it moves from F_i to F_j, as shown in Fig. 3b.

As the size and shape of moving t is undeterminable by a WSN and t is uncooperative, we use the location of its center of gravity to find t's location. It may move in a dynamic pattern. We tackle several moving patterns of t, as shown in Fig. 3c, Fig. 3d, and Fig. 3e, respectively: (i) straight moving, (ii) left or right turning, (iii) making U-turn. The moving direction of t can be presented by θ. Let time h be further divided into a series of discrete time instant, e.g., h_1, h_2, and at each such time instant, the interval is set to 1. Let $M_t = \{l_i, \theta_h, \theta'_h, v_t^{max}, v_t^{max'}\}$ be the motion of t at current time h, where θ_h and θ'_h are the direction angle and the turning angle, v_t^{max} and $v_t^{max'}$ are the maximal straight velocity and turning velocity, respectively. Each node can calculate the displacement of t denoted by D_h in the three patterns.

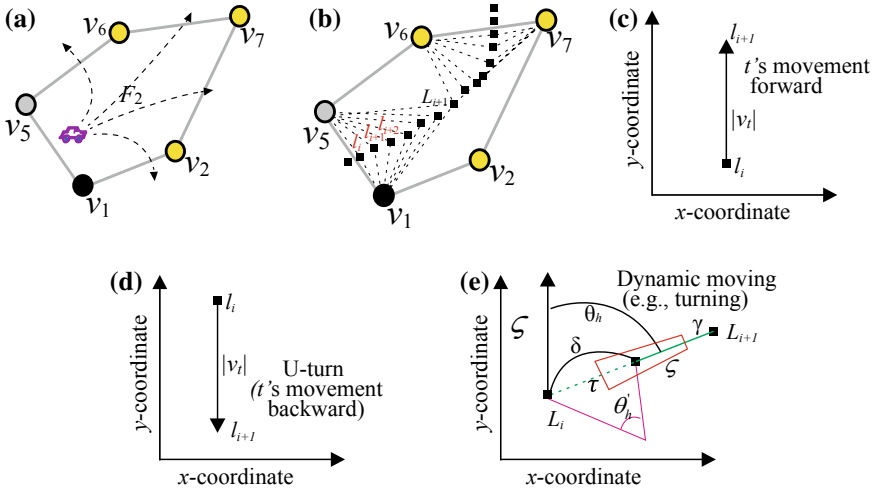


Fig. 3 **a** t may move in any pattern; **b** an example of localization of t by monitor and backup at each time instant and the moving sequence inside the face; **c**–**d** the models of t's different moving patterns (straight, U-turn, and a turning moving, respectively)

Pattern 1: $\theta_h = 0$, t moves in a straight pattern, i.e.,

$$D_h = v_t^{max} \tag{7}$$

Pattern 2: $\theta_h \in (-\pi, \pi]$ and $h_1 < 1$, t moves with turning angle θ_h'. Let h_1 be the time elapsed in turning in h. t's moving sequence consists of δ and γ, as shown in Fig. 3e, where δ is the arc formed by t's turning, and γ is the displacement that t moves in direction θ_h. Let ς be the length of the straight moving of t, we have the follows:

$$\begin{aligned} &\delta = \theta_h{}'\varsigma \\ &h_1 = \frac{\delta}{v_t^{max'}} = \frac{\theta_h{}'\varsigma}{v_t^{max'}} \\ &\gamma = v_t^{max}(1 - h_1) = v_t^{max}(1 - \frac{\theta_h{}'\varsigma}{v_t^{max'}}) \\ &\theta_h{}' = \frac{v_t^{max}}{\varsigma}\tan(\theta_h) \end{aligned} \tag{8}$$

Let τ be the chord length of δ, thus,

$$D_h = \tau + \gamma = 2\varsigma \sin(\frac{\theta_h{}'}{2}) + v_t^{max}(1 - \frac{\theta_h{}'\varsigma}{v_t^{max'}}) \tag{9}$$

Pattern 3: $\theta_h \in (-\pi, \pi]$ and $h_1 \geq 1$

$$\begin{aligned} \delta &= v_t^{max'} \\ D_h &= \tau = 2\varsigma \sin(\tfrac{v_t^{max'}}{2\varsigma}) \end{aligned} \tag{10}$$

Given the current location information of t at h, the predicted location l_{i+1} with approximate coordinates (x_{i+1}, y_{i+1}) of t at $i+1$ is given by

$$\begin{aligned} x_{i+1} &= x_i + D_t \cos\theta_h \\ y_{i+1} &= y_i + D_t \sin\theta_h \end{aligned} \tag{11}$$

By using (11), t's movement is determined by $(l_i, \theta_h, v_t^{max}, v_t^{max'})$, where v_t^{max} and $v_t^{max'}$ are influenced by the motion capacity of t. We can compute all possible locations of t by varying θ_h from $-\pi$ to π. We present Algorithm 3 to compute t's moving sequence based the above information (t's displacement D_t at a moving pattern and the estimation of a set of locations, see Fig. 3b). Based on t's movement sequence, we need to compute "face prediction," i.e., t's moving from one face F_i to another face F_j.

Algorithm 3: Computing Target Moving Sequence

Input: Given a target t, its previous locations l_{i-1};
Output: Target moving sequence inside face F_i;
1: Given the detected F_i via the Algorithm 2;
Set $M_t = \{l_i, \theta_h, \theta'_h, v_t^{max}, v_t^{max'}\}$;
2: **for** each further location measurement **do**:
Obtain a set of t's observations, $S' \leftarrow S_i, i = 1, 2, 3, \ldots,$;
Get $S_{med} \leftarrow$ the median$_i$ of the observation set S';
Compute D_t by S_{med} considering the three patterns;
Compute l_{i+1} by (11);
end for
3: Update $l_i \leftarrow l_{i+1}$ and buffer it;
4: Compute t's moving path sequence based on $l_i, i = 1, 2, 3, \ldots,$;

4.4 Face Prediction

In this subsection, we discuss the final stage of tracking, i.e., we observe t's moving from a current face F_i to a future face F_j, and propose Algorithm 4 for the prediction of face F_j. Through Algorithm 3, each monitor and backup can continue tracking t in its state s_0 and compute the moving sequence of t within R_s. Meanwhile, the distant neighbors of F_i are either aware of t or already involved in the tracking. Whenever t moves from F_i to F_j, it must leave the sensing areas of the monitor and backup and enter into the sensing areas of distant neighbors (see Fig. 4b). Based on the movement sequence, the monitor and backup compute a *probability of direction*, denoted by p,

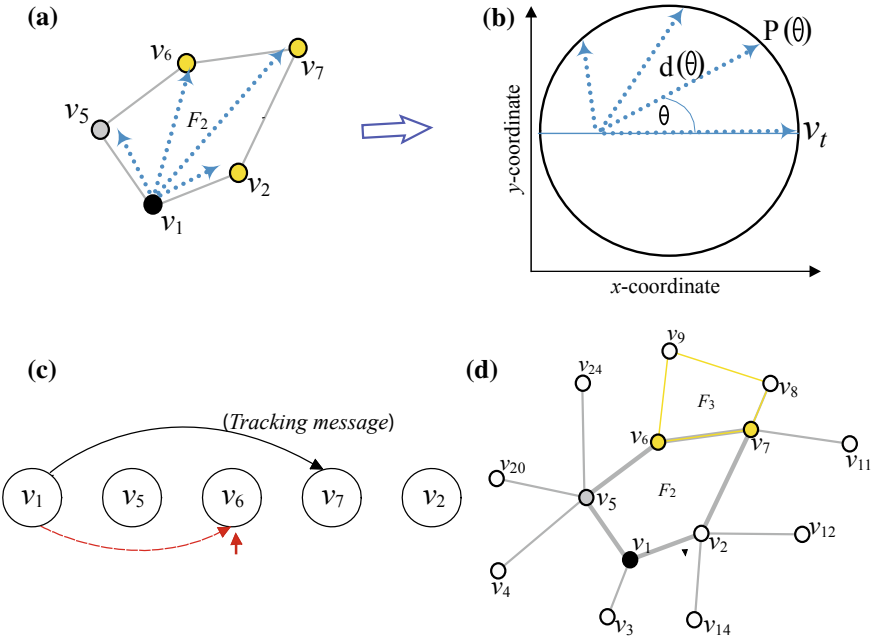


Fig. 4 The monitor v_1 in the face F_2 indicating t's moving in the direction from F_2 toward F_3

of the neighbors' locations, as shown in Fig. 4a–c. They predict t's moving toward the neighbors so that the neighbors can be further aware of t's moving from F_2 to F_3. Two of the neighbors (e.g., v_6 and v_7) become the new monitor and backup. They are the two common nodes between F_2 and F_3.

The monitor and backup use almost the same direction used in Algorithm 3 to calculate p. In the calculation of p, we consider the edge between the monitor and backup to be mapped over the x-axis in the 2D field, as shown in Fig. 4b. The direction arrow from node v_1 to node v_6 indicates that t is moving in the direction. θ_p specifies the direction, $\theta_p \in (-\pi, \pi]$. $\theta_p = 0$ is the instant direction at the current time. $d(\theta_p)$ and $p(\theta_p)$ are the radii from the monitor to the neighboring nodes in the face at direction θ_p and at the directional probability, respectively. p decreases linearly and reaches the minimum value at the direction of $d(\theta_p = \pi)$, and

$$\begin{aligned} \alpha &= x_{i+1}\theta_p + y_{i+1} \\ \alpha^* &= -x_{i+1}\theta_p + y_{i+1} \end{aligned} \tag{12}$$

The following equation is used to get the value of p:

$$p = \begin{cases} \alpha, \theta_p \in (-\pi, 0] \\ \alpha^*, \theta_p \in (0, \pi] \end{cases} \tag{13}$$

Algorithm 4: Face Prediction by the Monitor

Input: Given a face F_i where t is moving, $i = 1, 2, 3, \ldots,$;
Output: Prediction of face F_j where t may move;
1: Get t's moving sequence inside F_i by Algorithm 3;
Compute θ_p by analyzing t's moving direction;
Compute α, α^*, and then p;
2: Issue an alert message to distant neighbors of F_i (i.e., the new monitor and backup) located in t's moving path;
if (upon receiving the message) it is the requested node **then**
Start node organization into F_{i+1} and neighboring faces;
else
The node stays in awakening state s_1;
3: **if** upon detection of t and reception of the final alert **then**
Interact with the new backup; // $v_i \leftrightarrow v_j$
Continue computing t's moving sequence in within its R_s by Algorithm 3 toward F_{i+1} it belonged;
else Interact with the previous monitor and all of its neighbors for t's detection; // t may make U-turn or turning in moving
4: **if** t is detected in face F_{i+1} **then**
$l_i \leftarrow$ the new location;
$F_j = F_{i+1}$ // this face becomes the current face;
end if

After computing p, the current monitor issues the *alert* message on the direction with the monitor election information. As t leaves the monitor's sensing area, the monitor finally alerts the new monitor by handing over t's information in a message form. An example of sending an alert message in a possible direction can be seen in Fig. 4c. Once the new monitor receives the alert, some of the nodes of F_i (e.g., F_2) also reorganize themselves into predicted face F_j (e.g., F_3), as shown in Fig. 5d, by following the node organization rules described before. Upon receiving the alert, a timer h_{out} (will be described further) is started to meet in real time with two purposes: (i) to specify the timeout period in which a new monitor waits to track t when t moves across between itself and the backup; (ii) prevent situations of any irregular event happening or any wrong local information in face F_3 and in the neighboring faces before t's moving to F_3. The pseudocode of this face prediction is presented in Algorithm 4.

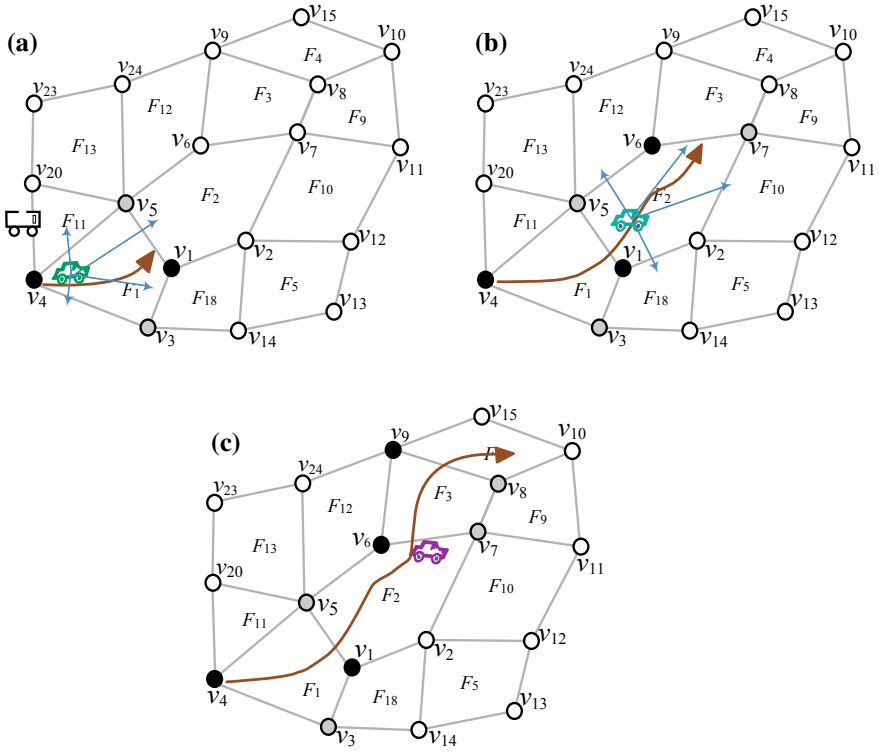


Fig. 5 Step by step target tracking through the faces in the WSN: **a** t is moving through F_1 and moving across the edge between v_1 and v_5; **b** t is moving through F_2 and moving across the edge between v_6 and v_7; **c** t is moving through F_3 and moving across the edge between v_9 and v_8

5 Tracking Process and Robustness

Based on the proposed tracking algorithms, we first provide the tracking process in this section. Then, we discuss the tracking robustness to failure events and recovery solutions.

5.1 The Tracking Process

This subsection discusses the interactions between T, the monitors and the backups during the tracking of t. The process is illustrated in Fig. 5. Once the WSN is deployed, the planarization is performed at the initialization. As T wishes, it sends a flooding request into the WSN at time h, requesting t' information. Since t is detected inside F_1, T is informed of t's current location, and the closest node is v_4 (see Fig. 5a).

v_4 becomes the monitor by using P_d and detects F_1. In F_1, v_4 has 3 neighboring nodes that are v_5, v_1, and v_3; its immediate neighbors are v_5 and v_3. When T needs information about t, T sends a query to the monitor. For reliable queries and replies, T uses Algorithm 5, which gives a snapshot of how T and monitor interact with each other.

Algorithm 5: Tracker T's Query

Input: Given a tracker T that moves throughout the WSN G;
T has a set of query message $Q = [q_1, q_2, \ldots, q_k]$;
Output: Reply to the answers to T by the Monitor and Backup;
T: send the queries with sequence numbers $1, 2, \ldots, k$;
$Monitor$: receive the message for q_k;
Check the sequence number for loss detection;
if a loss is detected in the sequence number **then**
Send an NACK to recovers the loss message;
end if
T: Check the sequence number;
if upon reception of the NACK **then**
Retransmits q_{k-1};
end if
$Monitor$: When the queries are successfully received
if received message is not the last queries **then**
The next query is not sent before timeout;
T retransmits this message until the ACK is received;
else
Monitor sends an ACK to T //receives the exact answer

If t moves in the direction of v_3, v_4 is able to easily determine it. As shown in Fig. 5a, if the monitor v_4 estimates that t is moving toward v_1, v_4 then interacts with v_1 and v_5. Since T arrives at the vicinity of v_4 after t has moved away, v_4 informs T. This means that v_4 will transfer the information to T to follow the same route, i.e., to go to v_1 (in the proximity of t's current location). The monitor interacts with T to transfer $t's$ information in order to capture t. Algorithm 6 provides the pseudocode of the interactions between the monitor and T for t's capturing.

When t moves from F_1 to F_2 along the route indicated in Fig. 5b, at time $h + 1$, only the monitor v_1 or both monitor v_1 and backup v_5 are already aware of t's route. Notice that when a monitor indicates the predicted location and direction of t where T will be after given h, it determines which node will most likely be the new monitor that t is approaching. It also elects an immediate neighbor of the new monitor as backup and informs them in advance about incoming t.

v_1 is already in s_0 and detects t as it moves from F_1 to F_2. v_1 receives the message from v_4 and forwards it in F_2. It then checks itself to confirm if it is the new monitor. As a monitor, v_1 senses and observes t in F_2. Meanwhile, T reaches the proximity of v_1's locations and requests $t's$ status (see Algorithm 6). Monitor v_1 works in the same way and elects a new monitor v_6 and a new backup v_7 by comparing with $t's$ future movements at time $h + 2$. Based on the tracking process above, we can find

Algorithm 6: Transferring t's Capturing Information

Input: A monitor and a backup that involves in tracking t in the WSN;
Output: Transferring t's information to T;
T: Move to v_i's proximity and query for t's updated information;
$Monitor$: Given a timer h_{out} for Monitor to send t's information;
Start h_{out};
Buffer the information until an ACK is received from T;
if $d(l_i^s, t) \leq \epsilon$ // t is still within R_s **then**
Send a message to T about t is in a capturing distance;
else
Send a message to the new monitor and backup
about $t's$ moving toward them;
Send a message to T
about directing it to the new monitor and backup;
Go to s_1;
T: **if** T reaches the monitor's proximity and h_{out} is not over **then**
Send a capture report to the monitor;
Stop tracking operation;
else if h_{out} is over or upon reception of the message **then**
Send an ACK to monitor about moving toward new monitor;
Move toward the new monitor;
end if
$Monitor$: **if** the ACK is not received by T **then**
Retransmit and reset time until t_{out};
else if upon reception of the capture report **then**
Go to s_2;
else
Go to s_1; //tracking operation continues

the secuences in the tracking. An example of a sequence of faces is $F_1 \rightarrow F_2 \rightarrow F_3 \rightarrow F_4$, and so on. An example of a sequence of locations of the monitor and backup is (v_4, v_3), (v_1, v_5), (v_6, v_7), (v_8, v_9), and so on. Both sequences represent t's movement sequence (or path) and T's routes.

5.2 Robustness to the Special Events During Tracking

5.2.1 Failure Event Definition and Identification

In this work, the following occurrences are specified and considered in order to prevent target detection failures or a loss of tracking:

- *Network failure* occurs when the monitor relays a message, but does not receive an acknowledgment in time. This is a result of losing an acknowledgment. A node may also fail during operation because of fault, energy depletion, or some other reasons.
- *Prediction failure* occurs if the prediction of the next location where t is approaching is incorrect, e.g., t may suddenly change its route or take a U-turn toward the opposite direction, which results in an inaccurate predicted location.

- *A connectivity hole* or *physical obstacle* is an area where the edges between nodes form a closed face without direct links between nodes that are not adjacent on the perimeter of the face.
- If *t*'s *detection failure* occurs, a monitor instantly issues a message to all face neighbors about changing their state, which may result in a short delay in tracking.

5.2.2 Recovery Solutions

`t-Tracking` usually avoids a single node failure that causes the partition of a sensor network. When a new monitor fails to detect or is not close to t, the backup takes up the role of the monitor. The relationship between the monitor, backup, and possible new monitor and backup is maintained by a low-cost, implicit linked list among them, as shown in Fig. 6a. When t comes across the monitor's area, the monitor constructs a linked list automatically and connects all the neighbors in the face, including the backup, immediate neighbors, and distant neighbors. There are linear links between the monitor v_4 and the new monitor v_1, and another link between v_4 and v_5. We also show a link between v_1 and its backup v_5.

If there is a node failure (e.g., v_7), the neighboring nodes perform a local maintenance (by merging two or more faces into one) around the failed node. In such a maintenance, the size of new face (e.g., F_{10} in Fig. 6b) become large. Thus, t can be detected inside the face. A similar solution is applied to if there is a permanent link failure due to an obstacle and hole in the WSNs, or other reasons. The monitor finds the optimal cycle (i.e., redundant link) to its neighbors. If the monitor can find one or more cycles, the current face is divided and merged with another faces into two or more sub-faces (see Fig. 6b). More details about such local maintenance on faces can be found in our earlier work [77]. Note that we do not insert any nodes inside a face unlike the work *Forms* [13]. We think that node insertion into a face during tracking operation is not suitable and is time- and energy-consuming.

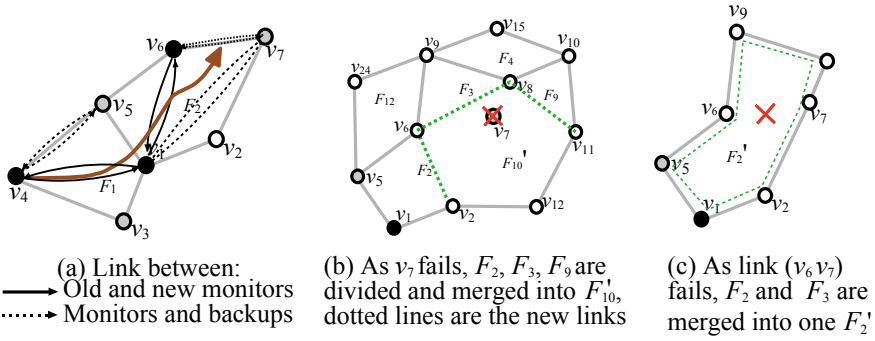


Fig. 6 Offering robustness to special events in the WSN locally: **a** link construction; **b** face dividing and merging; **c** face merging

If there is an obstacle (either a concave or convex obstacle), the face may not be divided because if message transmission is impaired by a physical obstacle, then the monitor alerts all of its face neighbors to be active and to detect t. During a discrete time interval, if t is not detected yet, the nodes in the neighboring faces become active and detect t. The link construction algorithm (see Algorithm 7) runs between the monitor and the face neighbors.

It also establishes the node connectivity of the assumed deployment characterized by the following:

- There exists a link between a monitor and a new monitor.
- There exists a link between a monitor and a new backup.
- Each monitor needs to connect to at least one node during the tracking time.

In order to discuss detection failure, suppose that v_6 is currently elected as the new monitor, and v_7 is the backup; if t is not in R_s of v_6, v_6 may fail to detect t. However, in `t-Tracking`, we consider that if v_6 fails to detect t, v_7 cooperates in tracking t. If v_7 also fails due to the fact that t may change direction or because of other failures, v_7 and v_6 send a detection failure message to v_1. Old monitor v_1 then sends a message to all of its neighbors in the face that tells them to cooperate in tracking t. These neighbors include v_5, v_6, v_7, and v_2, except itself in F_2.

Algorithm 7: Link Construction

Input: Given links of a current face F_i and L_i, cycle;
Output: Communication link list and re-cycle;
Step 1: (Further node organization into faces)
Connect backup and all other nodes (one by one);
Duplicate all of its edges;
if there is an edge from v_i to v_j **then**
Add a second edge from v_i to v_j;
Find a communication cycle in it;
Step 2: Check all edges are visited and cycled;
if an edge is not visited **then**
Construct a cycle;
if a cycle cannot be constructed **then**
Link all of the nodes through "perimeter" of the face;
repeat
Each next node ← get each next 1-hop (perimeter, neighbor)
(using right hand rule [6, 51, 52, 63]);
until the last node is the monitor itself;

In this case, v_1 is required to wait for a new monitor. If monitor v_1 gets a message about the presence of T, it replies with t's detection failure. If t is not detected in F_2, then v_1 sends a message to all of the nodes in the neighboring faces that correspond to its immediate neighbors and distant neighbors through them. After that, if t is not detected in the neighboring faces, monitor v_1 sends a request to T to relocate t. T sends a message to the WSN to relocate t. This mechanism mitigates the chance of routing failure, node failure, as well as the event of target missing. Based on t's

moving sequence, a face can be detected in advance so that if there is any hole or obstacle, the monitor is aware of it.

In the performance evaluation of t-Tracking, we found that this mechanism to detect t further by nodes in the neighboring faces is required for a very few times, specifically, two times in 50 simulation runs. Additionally, if t unfortunately stays in a hole region or t is not detected at all (due to an unknown reason), T gets a message about t missing. It means that T cannot get any information about the next destination. Then, if it can hear the nodes and receives a request after a predefined time instant, it relocates t. When t leaves the hole, t can be further detected in a face (outside the hole), and T can obtain its location. We found "no relocation" request to the WSN by T in the 50 simulation runs.

5.2.3 Handling Sparse or Dense Network Area During Tracking

The network density, denoted by ρ, is not bound in t-Tracking. When there is a constant bound on ρ (e.g., VigilNet [15], Forms [13]), we believe that our solution is highly feasible because we use different t's speeds while many holes may appear in the WSN. However, if ρ is large, especially if ρ is larger than a given threshold in some areas, fewer holes may appear, but t's moving sequence and face prediction cannot be estimated efficiently, and n and S_{max} can be high. The monitor is aware of the local network density. When v_5 corresponds to 5 faces, as shown in Fig. 5, we consider this as average density. When a monitor corresponds to many faces but the amount of nodes in each face is small, the WSN is dense. Conversely, when a monitor corresponds to a few faces but the amount of nodes in each face is still small, the WSN is sparse. In a dense WSN, the monitor relaxes the frequency of computing t's localization; i.e., after taking a few observations, the monitor estimates t's locations and predicts the next face directly. In that case, face prediction is preferred rather than a moving sequence of t. More concerns with the proposed tracking algorithms can be found in Appendix B.

6 Design of t-Tracking

In this section, we delve into the design t-Tracking. Then, we provide the state transition techniques and the detail of the energy consumption model.

6.1 Key Design Elements

The key design elements of t-Tracking scheme are presented in Fig. 7. The target node element is the mobile target that can emit various forms of signals and has mobility function. In addition to the mobility function, the tracker element is

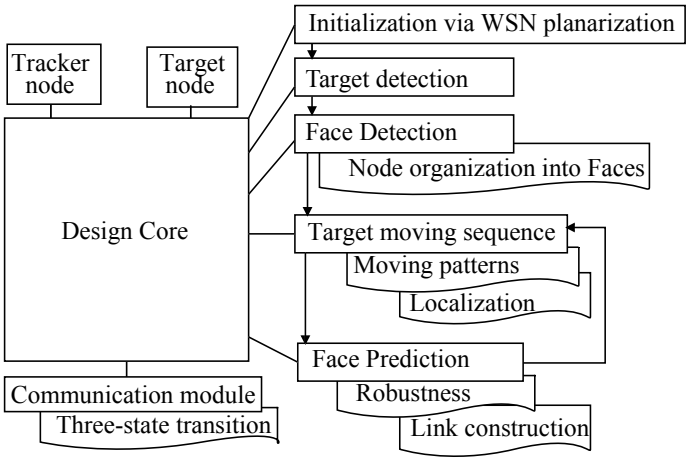


Fig. 7 Key design elements of `t-Tracking`

equipped with a communication radio to interact with the WSN. The communication element of a sensor node interacts with neighboring nodes and the tracker. The element follows the rules of energy-efficient state transition and its duty cycle. The WSN planarization is carried out after its deployment. Once the WSN starts tracking operation, t's detection element commences. Once t is detected by some nodes at the beginning through the "target detection" algorithm element, all of the nodes in the vicinity is organized into faces via "face detection" algorithm element. Two of the nodes become the monitor and backup. Then, the main tracking operation starts. In the tracking, the computation of "target moving sequence" and "face prediction" elements are carried out until t is captured. The design core element handles all the detail of tracking operations, including communication tasks, data recording and buffering, timing, synchronization.

6.2 Sensor State Transition Techniques and Energy Saving in the WSN

Putting sensors into an inactive state is the most widely used and cost-effective technique to save an amount of energy for future usage [78]. We present state transition techniques for sensor nodes and make it suitable for tracking through faces. A node has three different states of operation.

- Active state ($s_{k=0}$): when the node participates in tracking and interact with others, i.e., the monitor and the backup and some of their neighboring nodes in face F_i.
- Awakening state ($s_{k=1}$): when a node awakes at a predefined time and senses for a short period of time and checks whether it has a request or not.

- Inactive state ($s_{k=2}$): when a node is in the sleeping mode. Relatively distant nodes from F_i can be in the inactive state. More precisely, the rest of the faces of the network, where there is no target, need not perform any active communications to maintain tracking data and can sleep or go to in this low-power sleeping mode for extended periods. In `t-Tracking`, since all the movements of t are handled locally, then the distant nodes in the outside of the neighboring faces can sleep or go to the low-power mode to save energy without fear of interruptions.

Figure 8 gives a general idea of how nodes in the network operate. The most energy-efficient state is s_2, in which the sensor turns its service off, and the duty cycle is set to very low. We build up its efficiency on the localized nature generated by the dynamic behavior of t, requiring that only the nodes in t's proximity are fully functional (i.e., setting the maximum duty cycle (D_c)) to implement the tracking operation [78, 79].

A sensor in state s_2 periodically awakens at a predefined period and changes its state to s_1. A sensor can only change its state to s_0 and its communication channel awaken when it is in state s_1 and it is ensured that it is the closest to t. If it is the closest, it then sends a message to the current node in s_1. If the node does not change its state to s_0 at this instant, it has not found t and there is *no request* received; it changes its state to s_2 for another period of time. If there is a *wake-up request*, a node changes its state from s_1 to s_0. A node only changes its state to s_0 when (i) it is in s_1, and it is ensured that it is one of the neighboring nodes of the requested nodes in F_i, and (ii) it has a degree of P_d(detection probability). If it is one of the neighboring nodes, it also then sends a message to its neighboring nodes.

Assume that a tracking event is captured by a sensor node at some h_0, processing is finished at time h_1, and the next tracking event occurs at time $h_2 = h_1 + h_i$. According to the state transition diagram in Fig. 8, each state s_k has a power consumption P_k, and the transition time to and from the state is given by $\tau_{d,k}$ and $h_{u,k}$, respectively. With this state transition, we reduce the activation and deactivation delay during

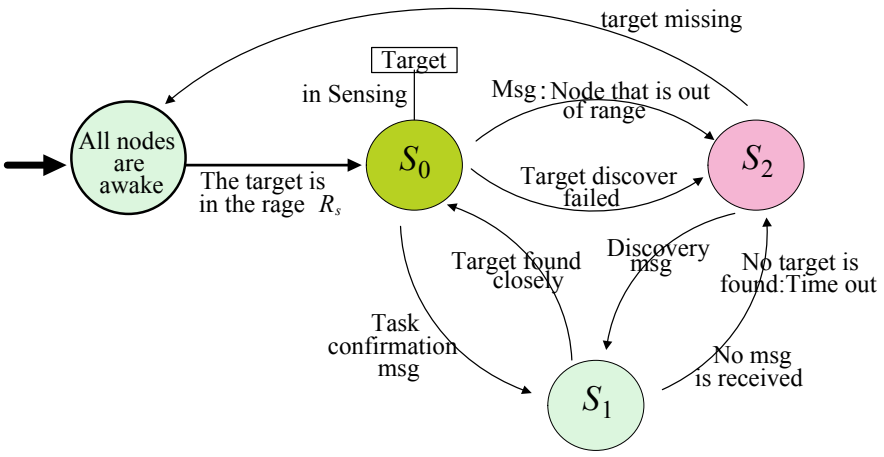


Fig. 8 State transition diagram with three-state model

the process of changing a state. There are several types of delays in tracking in WSNs. More details about the delays can be found in [15]. Typically, in different node states, $P_j > P_i$, $h_{d,i} > h_{d,j}$ and $h_{u,i} > h_{u,j}$ for any $i > j$, and $\Delta P = P_0 - P_k$ and $\Delta P' = P_0 + P_k$. When the node changes its state from s_0 to, say s_k, individual components such as the radio, memory, and processor are powered down with time. Utilizing this technique, the amount of energy saving during tracking is denoted by E_{s_k} because the state transition given by the difference in the face and sleep thresholds $T_{th,k}$ corresponding to the states is computed. The equation forms are as follows.

$$E_{s,k} = \Delta P h_i - \frac{1}{2}(\Delta P \tau_{d,k} + \Delta P' \tau_{u,k}) \tag{14}$$

$$T_{th,k} = \frac{1}{2}\left[\tau_{d,k} + \frac{\Delta P'}{\Delta P'}\tau_{u,k}\right] \tag{15}$$

This state transition, as shown in Fig. 8, takes advantage of the energy-saving feature in WSNs while the nodes, e.g., the monitor and backup, are in the active state one by one, and the other nodes typically stay in a periodic awakening or inactive state.

6.3 Energy Saving During Target Tracking

We calculate the energy consumption considering all aspects in tracking, including communication, localization, listening, and idleness. To save energy, we delve into an energy evaluation model for t's detection and positioning [26]. Conventionally, the radio transceivers are considered to have three power levels corresponding to sensing (P_s), transmitting (P_{tr}), and receiving (P_r) (see Table 3 for symbol description). In addition, we consider another essential state, which is almost ignored in many tracking applications, called idle power (P_l) (including waiting time to receive, wakeup delay, initial activation delay, and detection delay). P_l often consumes as much energy as the receiving level, even more energy than if the node would be asleep. This occurs when a node is listening to the channel, but is not transmitting or receiving any data. In `t-Tracking`, a node in s_2 agrees on a common T_p and a common D_c. For each h, the node wakes up randomly and stays awake for ($T_p.D_c$), then goes to sleep. Different kinds of delays and their effects are well described in [14, 15]. To achieve a good tracking system, the energy consumption of delays should be considered. We aim to reduce the idle listening time and to meet the real-time requirements.

Assume at a time instant, the number of nodes involved in communication and detection is denoted by $S_d(h)$, and the number of nodes that confirm the detection report of t is implied by $S_c(h)$. $S_d(h)$ are those nodes of faces that continue tracking after detection. They may be the monitor and backup and also may be the distant neighbors. $S_c(h)$ includes some of the nodes of outside faces that may be able to

Table 3 More notational conventions

Symbol	Description
P_s	The power level for sensing
P_{tr}	The power level for transmission
P_r	The power level for reception
P_l	The power level for idleness
$S_d(h)$	The number of detecting nodes that continue the tracking after detection
$S_c(h)$	The number of nodes that confirm the detection
E_m	The energy consumed for communication by the monitor
E_b	The energy consumed for communication by the backup or other nodes
E'	The energy consumed for the preceding t's localization and tracking
E_l	the energy consumed for low-power listening
E_{in}	The instantaneous energy consumption
T_m	The time required for querying a message
T_r	The reports for tracking information forwarding to the tracker
T_e	The reports for any communication events (face generation, tracking, etc.)
T_q	The reports for all queries in the WSN

detect t, but withdraw themselves from the tracking operation, due to the focusing on the current face (See Concern 1 and 2 in Sect. 7.4). Therefore, the energy for sensing activities in the wireless network is denoted by $E_s(S_d(h))$. E_m and E_b denote the energy used for communication by the monitor and the other nodes, respectively. The equation forms are as follows:

$$\begin{aligned} E_m(S_d(h), T_m) &= (P_{tr} + P_r)T_m\ S_d(h) \\ E_b(S_d(h), T_m) &= (P_{tr} + P_r\ S_d(h))T_m, \end{aligned} \tag{16}$$

where T_m is the time required for querying a message. In this work, the parameter T_m is directly proportional to the volume of messages exchanged in communication, including the total amount of data transmission (tracking report) and the total number of interactions (the communication for connectivity, fault tolerance, and so on). It can be one of three values: T_r is for the reports forward toward to the tracker/sink about t's tracking (detection and localization), T_e is for any kind of event reports exchanged in the WSN, and T_q is for query request. They satisfy the relationship $T_e \leq T_q \leq T_r$. It is considered that the target detection and localization are discrete processes derived from a discrete sampling of t's moving in the WSN.

In `t-Tracking`, t's moving in the sensor field during the time interval $[h_s, h_e]$, E_{in} denotes the corresponding instantaneous energy consumption, and E denotes the total energy consumption in the WSN. The energy consumption for low-power listening, E_l, is expressed as: $E_l = P_l(h)$. Hence, E is calculated at h as follows:

$$E_{in}(h) = E_{in} + E_l(h) \tag{17}$$

Thus,

$$E = \sum_{h=h_s}^{h_e} E_{in}(h) \tag{18}$$

We suppose that the maximum number of sensors in a face, or in neighboring faces, that are requested by the monitor to be informants about t is implied by $S_{\max}$, which is actively participating in locating t. Energy consumption is evaluated using the preceding t's localization and is denoted by E'. E' is given as follows:

$$\begin{aligned} lE^{'}(h) = {} & E_s(S_d(h)) + E_m(S_d(h), T_m) \\ & + E_m(S_c(h), T_m) + E_m(T_r) \\ & + E_b(S_d(h), T_m) + E_b(S_c(h), T_m) \end{aligned} \tag{19}$$

$$E^{'} = \sum_{h=h_s}^{h_e} E^{'}(h) \tag{20}$$

Let E_{tr} be the amount of energy savings achieved from all of the aspects during tracking. The amount of energy savings is the difference in energy consumption during t's localization and tracking stated as $E_{tr}(h) = E - E^{'}$. Hence,

$$E_{tr} = \sum_{h=h_s}^{h_e} E_{tr}(h) \tag{21}$$

Here, it is essential to point out that t's detecting sensor may be the new monitor, backup, distant neighbor, or outside neighbor which is required to confirm the monitor instantly as t is detected. It is known that t cannot be detected by all of the requested sensors. Thus, we can state $S_d(h) \leq S_c(h)$. This is sufficient for selecting sensors in order to save energy. With the proper selection of $S_{\max}$, where $S_s(h) < S_{\max}$, E reduces greatly as time goes on.

7 Performance Analysis

In this section, we conduct the design-related performance analysis.

7.1 Cost of Fault Tolerance in Detection and Tracking

There are plenty of central server-based (or sink-based) detection and tracking schemes that are usually required to keep most of the regions of the network on for frequent updating of t's information at a far away location of the sink, and

network-wide flooding and routing. Also, the update costs in a sink-based tracking scheme depends on the network size, which often requires a huge amount of messages per tracking event. This is because such a scheme requires forwarding the updates to the sink at several levels of hierarchy. Thus, it is not easy to update anything that occurs at a local area, such as the event of t's missing or sensor faults. A node has to flood all the nodes inside the radio range r_c. The communication cost of which is proportional to the area of r_c, i.e., $A(r_c)$.

Compared to such a sink-based scheme, the amount of data to be sent is small in `t-Tracking`, since the nodes in the proximity of t can detect t and can locally cache the detection data. During tracking, no flooding is involved in the faces. Although, we expect that t moves often and in dynamic patterns, there is never any need to send updates to a distant end node or node in the neighboring faces in normal cases of tracking. This is also significant from a sensor energy cost point of view. Since t's movements are detected and handled locally inside F_i and F_j, relatively distant nodes can sleep or go to a low-power inactive state to reduce energy cost.

Thus, a node has to issue/query messages to its immediate neighbors and neighboring node in F_i. The communication cost of which is proportional to the edges (on the perimeter) of faces in r_c (which is equivalent to e_i covered), i.e., $F(r_c) \ll A(r_c)$. Looking into more detail about cost in `t-Tracking`, if one or more nodes fails a F_i fail, this does not affect a great deal of QoT. More precisely, if either the monitor or backup fail, making a neighboring node as one of the monitor or backup to recover from the situation through network maintenance (i.e., merging faces) can be at a cost of $O(n_{F_i})$, where (n_{F_i}) is the number of nodes used in merging the faces. If more than one node fails (including the monitor or backup), the F_i can be merged to the F_j, or to any of the neighboring faces, at a cost of $O(F_i)$ to recover from the faults. In addition, the network incurs a cost in updating the tracking information as t moves from one face to another. Therefore, the total cost of the update equals the number of faces travelled by t. If updating edges between all neighboring nodes of each face requires communication between the monitor and backup, and the monitor and other neighboring nodes, then the update (communication) cost is also $O(n_n{}^* n_m)$. Here, n_n is the number of neighboring nodes used and n_m is the number of monitor and backup used.

In Fig. 9, an analysis is conducted on the performance results achieved through simulations with 300 randomly distributed sensors (for simulation settings, see Sect. 8.1). We inject faults into the WSN to investigate to what extent the update cost (for recovery from sensor node faults/failures) increases. Fault rate denotes the fault injection in a random manner after the planarized WSN initialization, estimated by the percentage of the number of nodes failed to the total number of nodes given. As t moves on the sensing field, one or more nodes (of some faces) in the WSN may be appeared failed.

It can be perceived by the results, as shown in Fig. 9 that, the update cost of a single tracking event can go into several hundred messages in a sink-based scheme (e.g., Forms, VigilNet). `t-Tracking` never has to update more than e_i edges, which requires an average of 36 messages per tracking event. Even better, when $n_m = 3$, i.e., if there are three nodes in F_i and one fails, another two become the monitor and

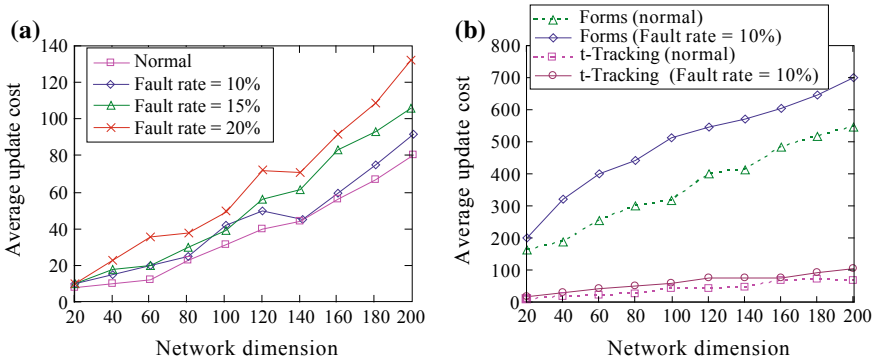


Fig. 9 The costs of fault tolerance in target tracking: **a** average update cost per tracking event in `t-Tracking` (under different fault rates); **b** average update cost per tracking event in `t-Tracking` and the Forms

backup, resulting in a lower update cost on the fault tolerance. There is also another reason that the local fault tolerance in `t-Tracking` reduces the chance of report loss on the way to T. When a node fails, the update cost in `t-Tracking` increases from 20 messages to 45 messages (32 messages on average), while the update requires one to several hundred messages in the sink-based scheme. In the case of more than one node failure, the situation becomes more serious. This means that the fault tolerance (local maintenance in `t-Tracking`) does not bring a significant cost on the WSN.

7.2 Wakeup Delay

Capturing delays are induced by a sleeping sensor in a face where t is approaching, because the sensor that cannot be woken up in time could result in average detection delays (AD). Hence, there can be a tradeoff between the saving energy and the tracking errors that result from the sleeping sensors. It could be even unreliable and difficult to tackle if the node, which is expected to wake up upon a request and respond to the requester in time, never wakes up because of fault or other reasons. On one hand, we successfully overcome these problems by modeling the energy-saving state transition and by reducing different kinds of delays which lead to the wake-up delay. On the other hand, a small wake-up delay cannot affect QoT because of the prediction of t approaching from one face to another. The sensing coverage is guaranteed that at least one of the nodes would turn into s_0, can sense and detect t as t appears in the area, and then find the backup. Thus, this tradeoff is optimized.

7.3 Sensing Task and Complexity

t's movement model in `t-Tracking` is simplistic but allows us to investigate sleeping policy for sensor. We just tried to keep our system simple and as accurate as possible. The sensing model presented is distributed in nature and can be executed by organizing the nodes into faces in order to follow t at each time step of tracking. Assume r is the number of samplings and ψ is the dimensionality of each S_i. Therefore, the algorithm has a linear complexity of $O(r\psi)$. Once a tracking simulation run, the incremental updating and getting new information for tracking only needs a few fractions of the computations. It is significant to mention that there is no need of target rediscovering in normal tracking. If there is a failure, it requires slightly more computation and communication costs than normal tracking.

7.4 Relative Distance

Within each period of tracking, a current monitor elects a new monitor. The monitor in that period determines the largest possible detection magnitude. As an alternative, the monitor can wait for some time for additional readings if t still exists in its R_s. The new monitor is not elected in this case, which would still keep the same destination for T to go to. In this respect, m is reduced and t can be captured quickly.

Concern 1 *How well can T capture t when t is far away from the monitor?*

In response to this concern, we need to know that how far T's location L_i is from the proximity of the monitor location l_i^s is determined as T receives a message that was sent by the monitor, which is related to the number of hops and faces away from the monitor. We do not assume that how far T's location is, but we think that T's speed can be faster than normal or t's speed at the time instant and T can take shortest path. T's location can be outside of the network area, but it is assumed to be in or near the network area at the system initialization. We work with calculating m that has a length varying from three to 12 hops. This implies that T's location can be about two to eight faces away. If T is either far away from or near to the monitor, T moves toward the monitor's location after getting a response. T's speed can be higher than normal.

Concern 2 *What happens when T fails to capture t at a capturing point?*

When T reaches the monitor location, it queries whether or not t is still in the capture distance. If t is with the distance, T can capture t. However, the capturing point should be inside the sensing area of the monitor, i.e., $d(l_i^s, l_i) < \epsilon$. When T reaches the monitor location l_i^s and t is in the boundary location (i.e., $d(l_i^s, l_i)$ is close to ϵ), T does not attempt to capture t at this capturing point. We leave it for the next capturing point. T is supposed to receive a message about the new monitor location if t goes away. Therefore, the distance $d(l_i^s, L_i)$ between T and the new monitor

is minimized as time goes on. We need to note that in most of time T is near or has reached the monitor location l_i^s in a timely fashion, but it has failed to capture t because of t's dynamic moving patterns. Then, T looks for the next capturing point. In this respect, when T spends more time to capture t, the energy consumption can be slightly increased.

Note that, particularly if `t-Tracking` is applied in other applications of WSNs, whether we can use the idea of target capturing or not, it does not matter, but we can follow a mobile entity (such as vehicle, animal, people, soldier, or intruder) and gather information about it.

8 Simulation Studies

8.1 Methods and Parameters

In this section, we evaluate the performance of `t-Tracking` through extensive simulations and provide more insight into the tracking issues from simulation perspectives. We implement it on the OMNet++ simulation environment using the Castalia simulator. We also utilize the extension of OMNet++: the Mobility Framework (MF) http://castalia.npc.nicta.com.au/index.php. We set up a basic network of 350 m $\times$ 350 m, where 300 nodes are randomly deployed. Roughly, 20 m away from each other, the nodes are deployed in a random manner with an approximately uniform density. The nodes communicate with each other by broadcasting messages across channels. These are assumed to be symmetric. We set $R_c \geq 2R_s$, which is to ensure that two nodes with an overlapping sensing area are capable of communicating directly with each other.

In the simulation, we adopt several system-wide parameters that directly affect the real-time properties in tracking t. The monitor, backup, and all the nodes follow the time synchronization with respect to each other. We set a millisecond-level synchronization to coordinate the operations among the nodes. To compute different types of delays, the nodes synchronize with T within 1–5 ms using the techniques presented in [15]. We adopt a number of parameters and values to see the performance of `t-Tracking`, which are similar to parameters in the real settings of VigilNet [14, 15].

The important parameters used in `t-Tracking` are listed as follows: (i) v_t is from 0.5 to 10 m/s, while v_T varies between 2, 6, 8, 10, and 12 m/s; (ii) the physical delay in target detection; (iii) the monitor node duty cycle (50%); (iv) the inactive duty cycle (0.5%); (v) the awakening duty cycle (5%); (vi) the required degree of aggregation (1%); (vii) $R_s = 20$ m (approx.); (viii) $R_c \geq 40$ m; (ix) the required number of reports for tracker $O(1)$; (xi) $\rho = (1\text{–}2.5)/400\ \text{m}^2$; (xi) $\epsilon = 5$ m. We model each sensor with six discrete power levels in the interval $\{-10\ \text{dB m}, 0\ \text{dB m}\}$, as the power settings of Imote2 sensor platform is tuned within the IEEE 802.15.4. The amount of energy consumption required by the levels is between 11.2 and 17.4 mA.

In most of the simulation configurations, we repeat the simulation for 200 times for high confidence and record the average as the final result. The energy consumption model and energy saving in different states of operation are similar to the ones described in Sect. 5.

Metrics. We consider the following metrics to evaluate the effects of those parameters on the system performance[2]:

- The total capturing time [C_T].
- The energy efficiency of the WSN [ΔE]: $\Delta E = E_{s_k} + E_{tr}$.
- The quality of tracking [QoT]: Suppose that t is moving in a sequence corresponding to e events (detection, localization, etc.) of the sensors' duty cycles (D_c). The rSTS (rate of successful tracking steps) divided by e is the QoT, which reflects the accuracy of tracking performance. The QoT can show how successful a system is against all the difficulties, such as the presence of high localization errors, low detection probability, sensor faults, physical obstacles, etc.

Before analyzing the simulation results, it is important to mention that here we discuss only the key results (focusing on the three matrices). Detailed results to support the key results (e.g., robustness to localization errors, detection probability, detection delay, performance on face detection) are discarded.[3]

Comparison. A high-level comparison between `t-Tracking` and three prior prominent schemes (namely, VigilNet [14, 15], Forms [13], and iTOA [35]) are performed by simulations. The choice of VigilNet is due to the similarity of its concepts of real-time design, its fast target tracking, and its sensor duty cycle. In addition, it has a real implementation through a physical test-bed with 200 XSM motes. Another choice is its system properties, such as energy consumption, accuracy (in terms of various types of delay measurements), and the use of sentry nodes (like the monitor in our work) in tracking. The choice of *Forms* is that the use of the idea of planarization and faces in tracking. The choice of *iTOA* is that the use of a mobile sink navigation, a localization method through TOA and a prediction method for tracking t.

8.2 Key Simulation Results

We begin by observing the performance of `t-Tracking` in Fig. 10 in terms of t's capturing time C_T. Figure 10 depicts the average required capturing time at T's different speeds in `T-tracking`. T can capture t at the 175th second (maximum time) and at the 37th second (minimum time). These results are compared to existing schemes, as shown in Fig. 10.

We calculate C_T from the system run to the time when t is captured. We consider the capture distance $\varepsilon = 5$, which is equivalent to $\frac{R_s}{4}$. We can see from Fig. 10 that VigilNet takes the longest time in capturing t. Although it reduces the detection delay,

[2]More metrics are considered in the Appendix C.

[3]Further results relevant to the three metrics are given in the Appendix C.

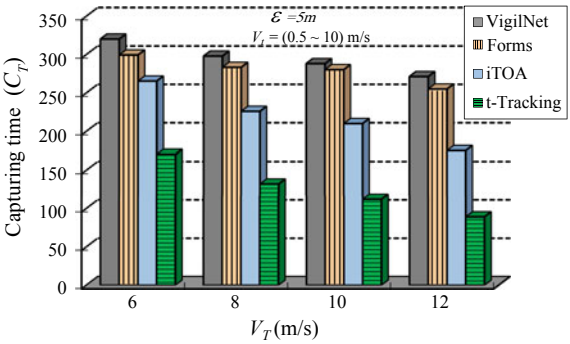


Fig. 10 The performance of the target capturing: average required capturing time T's different speeds in different schemes

the events of target missing are higher (due to strictly depending on the localization accuracy and other reasons) than all other schemes. ε is required to be higher in both VigilNet and Forms than in both iTOA and `t-Tracking`. In an investigation, we found that when we increase ε to 10 m and to 15 m, VigilNet and Forms still provide low capturing points.

When $v_t \geq v_T$ and $\varepsilon \leq 10$ m, it is difficult for T to capture t. Thus, the frequency of events of target missing is larger in iTOA and VigilNet than `t-Tracking`. `t-Tracking` achieves a superior performance in capturing t: T can capture t faster (from 52 to 92%) than all other schemes. In most of the times, the capturing point is inside R_s of the monitor and the backup, where $\varepsilon = 10$ m. For example, as shown in Fig. 10, when $v_T = 8$ m/s, T can capture t at the (158, 132, 103, 88)th second and the link in different simulations of `t-Tracking`, while these are the (257, 218, 201, 157)th second in iTOA, and the (318, 283, 272, 251)th second in VigilNet.

Next, we report the energy consumption (E) of the WSN after t is captured, as shown in Fig. 11a. In the simulations, we observe t's moving with a changing speed. We change v_t from 2 to 10 m/s. We can see that when v_t is between 6 and 10 m/s, the reduction in energy consumption of the WSN is drastic in `t-Tracking`, while it is gradual in iTOA and typical in both VigilNet and Forms. Figure 11a reveals that Forms consumes more energy than VigilNet and iTOA. Forms uses a large number of nodes in each tracking steps more precisely; n is 18–35 and S_{max} is 13–19 (on average) in Forms, while n is 11–15 and S_{max} is 7–10 in VigilNet, and n is 10–14 and S_{max} is 6–10 in iTOA. Although delays are handled comprehensively in VigilNet, the tracking still consumes a lot of energy. Many nodes, including sentry nodes, are used; the final decision about t's detection needs a large amount of communication (including additional interaction between the nodes and the sink). Overall, VigilNet and iTOA take robustness actions in a repetitive manner to further discover t, once there is a large amount of localization errors or the events of target missing. When v_T increases, the average energy consumption decreases in VigilNet and iTOA, but the events of target missing increase.

In contrast, n is 5–10 and S_{max} is 2–6 in `t-Tracking`. In most steps of tracking (about more than 70%) in all the simulation cases, $S_{max} = 2$–4. Unfortunately, if t

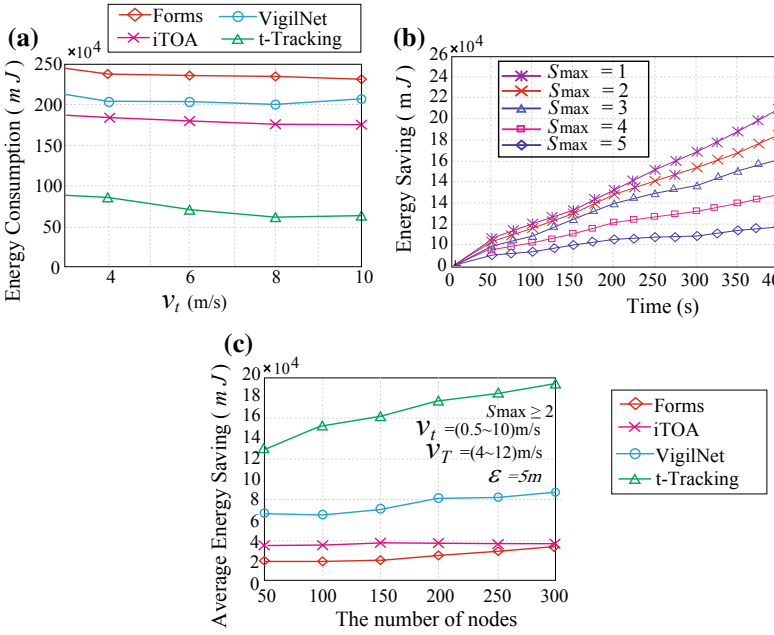


Fig. 11 Network performance: **a** average energy consumption versus v_t; **b** cumulative energy conservation as time goes on in `t-Tracking`; **c** average energy efficiency in all schemes

is not in R_s of the monitor and backup, it queries all of the neighbors in a face or in neighboring faces. In such a cases, S_{max} increases. More importantly, the monitor and backup are only used to interact with T in tracking t. Thus, fewer nodes, less local interactions between nodes, less interactions between the tracking nodes, and low capturing time help in reducing a significant amount of energy consumption. Figure 11b, c depicts the energy efficiency ($\Delta E = E_{s,k} + R_{tr}$) as time goes on in each simulation in `t-Tracking` and energy saving versus the number of sensor in the WSN in all the schemes, respectively. In both figures, the energy saving is averaged by the data gathered from 200 uninterrupted simulations with different parameters. ΔE is significantly high in `t-Tracking`, which is about 4–6 times improvement on the energy saving compared to other schemes.

Finally, we study of the QoT in `t-Tracking` and in other schemes. We analyze rSTS based on the overall simulation results, considering the mentioned underlying difficulties. Figure 12a shows that the QoT varies according to v_t. We observe that when n is very small or large (sparse or dense), the QoT is slightly lower than when n is medium ($3 \leq n \leq 6$). The QoT is fairly close to the maximum (0.8) in `t-Tracking` when $n \geq 4$, v_t is (8–10) m/s, and v_T is (6–8) m/s.

Figure 12b depicts that the QoT in different schemes. n and S_{max} in both Forms and iTOA are at least 3. When $n = 4$, QoT is around 0.55 in iTOA, 0.57 in VigilNet, 0.68 in Forms, while it is 0.75 in `t-Tracking`. The QoT is close to 0.8 when

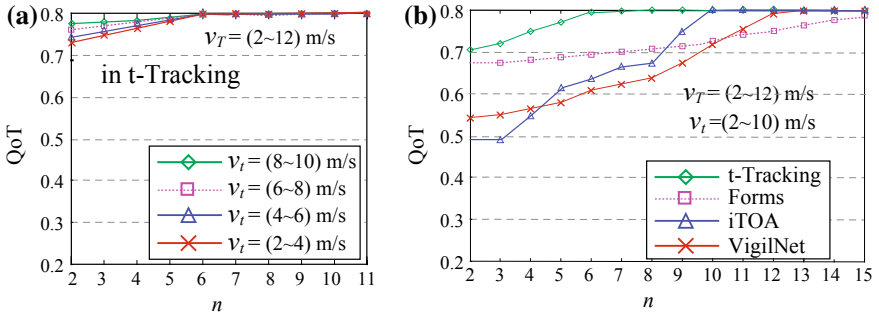


Fig. 12 The performance of QoT: **a** QoT versus n (the number of nodes used in each step of tracking) in `t-Tracking`; **b** QoT versus n in different schemes

$n = 18$ in Forms, $n = 12$ in Vigilnet, $n = 10$ in iTOA, and $n = 7$ in `t-Tracking`. This reveals that `t-Tracking` achieves higher QoT than prior schemes, which is about 42% higher than VigilNet, 35% higher than iTOA, and 27% higher than Forms. Both VigilNet and iTOA performs worse than `t-Tracking` and Forms, as VigilNet and iTOA heavily rely on high localization accuracy. Once there is an event of target missing caused by a large localization error or by a node failure, discovering and localizing t further is difficult. Besides reducing the dependency on localization similar to Form, we considers t's different movement patterns and face prediction in real-time in `t-Tracking`. This is why, it has superior performance on the QoT than that of Forms and others.

9 Proof-of-Concept System Implementation

In this section, we describe the evaluation of our proof-of-concept system implementation and analyze the obtained the results. experimental results on our objectives.

9.1 System Setup and Parameters

The system is deployed in the field of our university campus stadium to track an experimental vehicle, which contains 20 Intel Imote2 sensors, as shown in Fig. 13. Each Imote2 sensor runs on TinyOS 2.0.1 [80]. We configure each Imote2's PXA271 processor to operate in a low-voltage (0.85 V) and low-frequency (13 MHz) mode [81], as the processing of signal energy emitted by the vehicle does not require a high CPU speed. PXA271 has 6 frequency and power levels as specified in [81]. The Imote2 is equipped with an CC2420 IEEE 802.15.4 radio transceiver from Texas Instruments. The radio supports a 250kb/s data rate. We adjust the communication

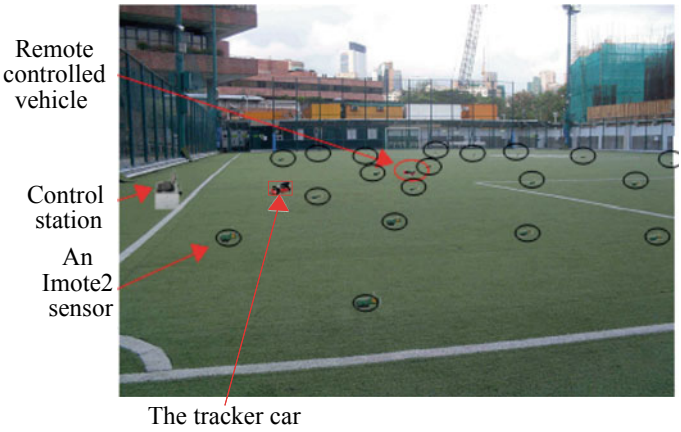


Fig. 13 Deployment site at the university campus stadium, where sensors, target car, tracker car, two experimental cars are deployed

range by adjusting the power level. It can communicate reliably up to 5 m at the lowest power level, but the interference range could be higher.

For the sake of simplicity, we mark the mote locations during the deployment in order get T's expected moving direction. A programmable remote control car equipped with a speaker (as a sound source) is employed as a mobile target t in the field. t is moved in dynamic patterns. Another programmable remote control car (comparatively bigger in size than the employed t) is employed as a tracker T. We place an additional Imote2 sensor as the sink node and a laptop on the car. The tracker car carries the laptop and the sink Imote2 that collects information from the WSN. The sink Imote2 is connected to the laptop via a USB port and transfers the collected data to it. Cygwin and Java virtual machines are installed on the laptop. Cygwin is used to configure the WSN. Through its command-based interface, applications are installed on the Imote2.

`t-Tracking` operates synchronously. We implement an implicit acknowledgment mechanism at the communication layer for per hop reliability in faces. The forwarding of a message acts as an acknowledgment for the sender. If an acknowledgment is not received, then messages are retransmitted up to 3 times (See Algorithm 5). The sink application is also invoked from this software. The size of all packets is 40 bytes, which includes 20 bytes for the packet header.

Based on the received t's moving patterns and the directions from a detecting node (a monitor or a backup) toward another nodes of a face of the WSN, the tracker car is controlled to move toward the specific node locations. It is moved in a straight, left, and right patterns to follow t in order to capture it. There is some tracking delays occurred by manually controlling the T's movement (controlling errors), during the upload and download time, and delay in processing at the laptop. One t is detected first time, the nodes require to set an *id* as a unique identifier of a respective t.

We configure $v_t = (0.1–3.5)$ m/s, $v_T = (0–3.5)$ m/s, $R_s = 8$ m, $R_c \geq 16$ m, $\epsilon = 2$ m. We repeated the experiments for a total of 20 rounds in two different days. We deploy the sensors on the second day with the intention to observe whether or not a variance in sensor locations have an impact on the system performance. We analyze the experimental results gathered from all of the rounds.

With no built-in sensors, Imote2 can minimize its power consumption and maximize its supporting flexibility when acting as a base to customized sensors. Building the sensor array becomes relatively straightforward on top of Imote2. We choose the widely used Panasonic WM-61B an as acoustic sensor. WM-61B is an omnidirectional condenser microphone that delivers acceptable sensitivity at a very low cost. The audio amplification circuit with the gain of 200 is designed around an LM386 op-amp (i.e., a very high-gain amplifier). The microphone array consists of four circuits.

We employ a small vehicle for t, which is equipped with a remote control module. An attached speaker, used as a sound source within the vehicle, is connected to the circuits and is tuned by the remote controller. When testing the data transfer in the ADC and the Imote2, we use the waveform generator in order to eliminate the noise factor from the microphone array. With this implementation, we conduct a variety of experiments where the localized face organization techniques are used as the underlying network interaction method.

9.2 t's Moving Traces

We now describe how the vehicle target t's traces are obtained. Imote2 sensors are deployed in a deterministic manner with about 12–16 m spacing at the field. We have moved t randomly in the operational field, and at the same time, we have computed t's coordinates using the Imote2. Sensors through face region traces are collected for when t is moving through the experimental WSN at different orientations. The monitors and backups of different faces are selected as t moves through the faces. The nodes of the WSN are organized into face regions with zero crossing edge. Crossing between a monitor and backup implies moving from one face to another face. The sensors use an *id* as a unique identifier of t so that other t (if there is any) and the tracker car do not bring any interference for the nodes in detecting the identified t. Two examples of tracking t can be seen in Fig. 14. In Fig. 14, each marked small circles indicates t's crossing between a monitor and a backup (or the common edge between two faces) in the WSN. Each movement, called a track, is of the form (timestamp, location) in the WSN. These tracks of t are then converted to tuples of the form (node *id*, timestamp, location, face sequence), where the face sequence is counted by the common edge between faces. Thus, by using the real traces collected from the field, we emulate t's detection, movement prediction, and then face detection, to evaluate the performance of `t-Tracking`.

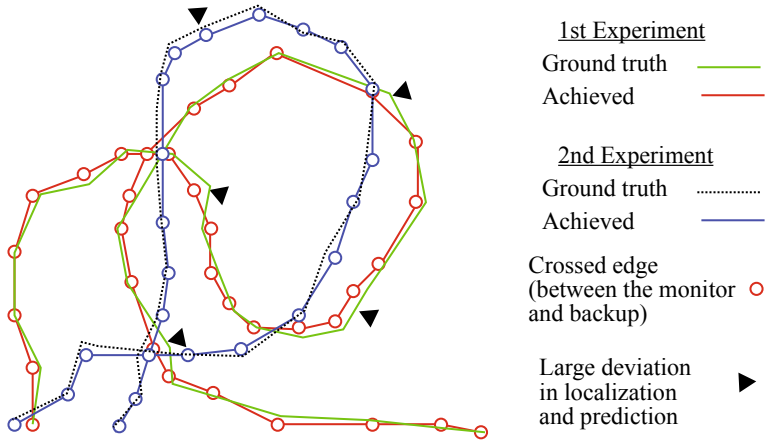


Fig. 14 Experimental tracking performance (face sequence) in two different runs of the experiment

9.3 Experimental Results

Figure 14 shows the trajectory (or face sequence) of t for two different runs of the experiment. The ground truth and achieved truth are represented by the black lines (one solid and one dashed) and by the red lines (also one solid and one dashed), respectively, which show the trajectories of t. The circled markers placed on the red line tracks correspond to the actual (ground truth) t's detection and movement prediction through the faces, computed by the Imote2s. The reason for showing both the trajectories with the dashed lines and the markers on these lines is to give a sense of when the system loses track of t. Especially, in the case that there is a large deviation in the achieved localization and movement prediction from the ground truth.

We report the performance of energy consumption (E) of the experimental WSN at a capturing point before t is captured, as shown in Fig. 15a, and after t is captured, as shown in Fig. 15. We take t's variable moving speeds and moving patterns into consideration by using t's moving direction probability, mentioned in the model section. In the experiments, we observe t's moving with a changing speed as shown in Fig. 15a. We change v_t from 0.5 to 3.5 m/s. We see that when v_t is between 1.5 and 3 m/s, the reduction in energy consumption is drastic in `t-Tracking`, while it is gradual in Forms, and typical in VigilNet. The energy consumption is higher in iTOA when v_t is 0.5–1.5 m/s, but is lower when v_t is from 2 to 3 m/s. It reveals that `t-Tracking` consumes low energy during t's capturing in practice.

One of the reasons for having a lower energy consumption in `t-Tracking` is the use of fewer sensors and less local interactions between sensors. Although delays are handled comprehensively in VigilNet, the tracking still consumes a lot of energy. Many sensor nodes, including sentry nodes, are used; the final decision about t's detection needs a large amount of transmission. The underlying information is that

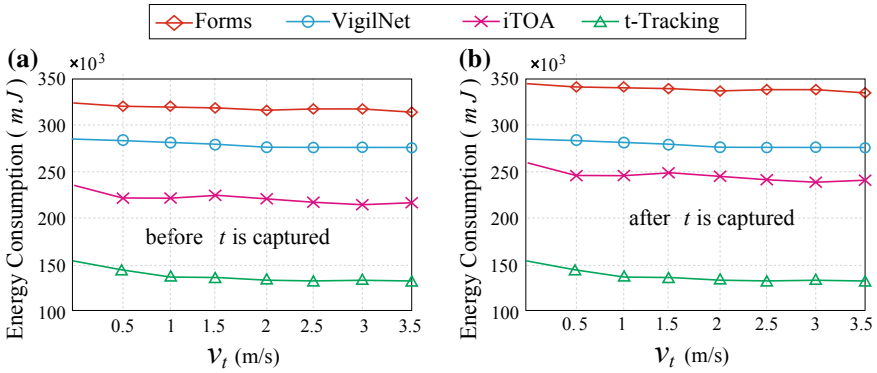


Fig. 15 The performance of energy consumption of the WSN: **a** before t is captured at a capturing point; **b** after t is captured at the capturing point

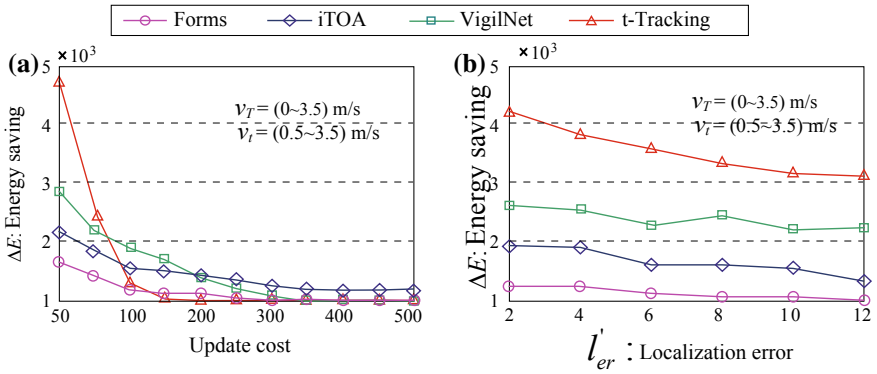


Fig. 16 The performance on the energy efficiency of the WSN: **a** average energy saving versus update cost; **b** average energy saving versus l'_{er}

in target capturing, much more time is needed in VigilNet than in `t-Tracking`. In both Fig. 15a, b, the energy consumption is averaged by the data gathered from experiments with different parameters. When v_T increases, the average energy consumption decreases in `t-Tracking`.

Finally, we discuss the energy efficiency of the WSN as shown in Fig. 16a. illustrates the energy saving versus update cost and Fig. 16b illustrates the energy saving versus localization errors in tracking. The update cost is between 350 and 400 in VigilNet and iTOA, which is less than Forms. The update cost in `t-Tracking` is around 150–200. Due to the lower update cost, the energy saving in `t-Tracking` is significantly lower than the other schemes, which is similar to results achieved in simulations.

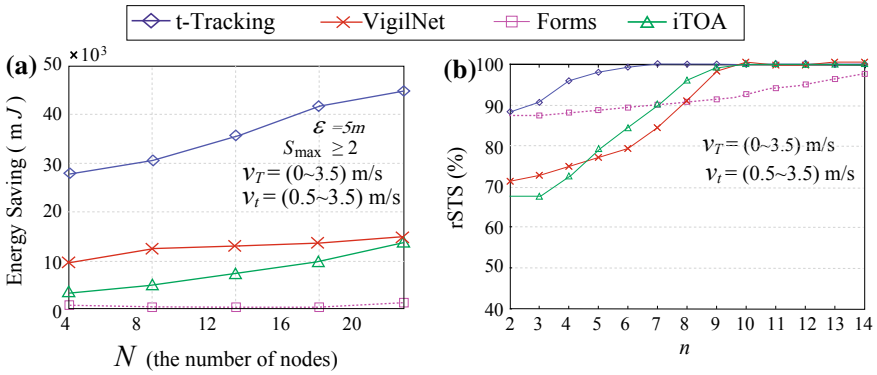


Fig. 17 Experimental performance: **a** energy efficiency in different schemes; **b** QoT versus n

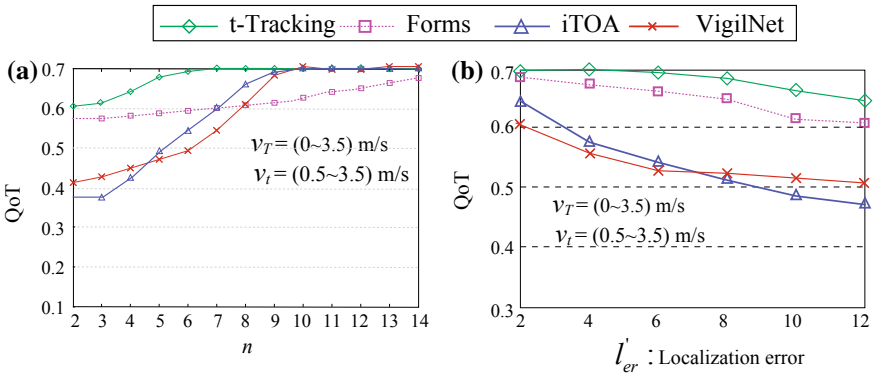


Fig. 18 Experimental performance on QoT: **a** QoT versus n; **b** QoT versus localization errors

The performance on energy efficiency also can be seen in Fig. 17a, where `t-Tracking` achieves at least five times improvement on energy saving compared to other schemes. One of the advantages is that the number of tracking nodes (mainly the monitors and backups) in `t-Tracking` is smaller than other schemes. Figure 17b demonstrates that the rSTS versus n. The rSTS reflects the QoT. We estimate QoT based on the experimental results, taking the localization errors and target missing into account. We observe that as n increases, rSTS increases. It also reveals that rSTS is more than 90%, when $4 \leq n \leq 6$ in `t-Tracking`. In many cases, $S_{max}(\leq n)$ is 2 and 3 in `t-Tracking`, while it is 3–6 in iTOA, 2–8 in VigilNet, and 3–14 in Forms. To get rSTS = 100%, VigilNet uses 9 nodes, iTOA uses 10 nodes, and Forms uses 17 nodes, while `t-Tracking` uses 6 nodes. The results show that `t-Tracking` outperforms all of the schemes, by achieving a higher rSTS (or QoT).

Figure 18a demonstrates that the QoT versus n. We estimate QoT based on the achieved rSTS and the event occurred in sensors' duty cycle. We can see that as n increases, QoT increases. It reveals that QoT is more than 0.6, when $3 \leq n \leq 6$ and

is 0.7 when $n \leq 4$ in t-Tracking. This validates our simulation results. It is often the case that QoT is more 0.6 when $S_{max}(\leq n)$ is 2 and 3 in t-Tracking, while it is 3–6 in iTOA, and 2–8 in VigilNet, and 3–14 in Forms. t-Tracking outperforms all of the schemes, by achieving a higher QoT (more than 60%) than them.

In Fig. 18b, when the localization error, $l'_{er} = 2$, the QoT is about 0.64 in iTOA and 0.61 in VigilNet. But, the QoT reduces drastically as the localization error increases in these two schemes. When there is a large localization error, $l'_{er} = 12$, the QoT gets down to 0.51 in VigilNet and 0.47 in iTOA. Both Forms and t-Tracking achieves higher QoT than that of VigilNet and iTOA. It is evident that t-Tracking performs better than Forms, VigilNet, and iTOA in the presence of large localization errors.

10 Conclusion and Future Work

In this chapter, we have proposed a new tracking scheme in WSNs, t-Tracking. Its ideas such as (i) fast tracking operation through face detection and prediction, (ii) reducing the interactions between a mobile tracker and monitor/backup, (iii) reducing the dependency on the need of strict localization accuracy in tracking can all be useful to many surveillance applications. Particularly, those applications that require keeping an eye on a mobile entity and pursuing the entity for different purposes such as safety, investigation. t-Tracking can track the entity by accomplishing a cooperation between the sensor nodes and tracker in a real-time manner and by requiring very little processing and communication energy requirements on the nodes. Furthermore, throughout our design, we exploit some physical properties of WSNs to achieve a functional, simple design that is robust to failures. For example, failure of one or a few nodes does not affect the operation of the network during t tracking.

Evaluation results obtained through extensive simulations and a proof-of-concept system implementation showed that the proposed scheme can highly contribute to energy conservation and quality of tracking in comparison to the recent prominent schemes. To the best of knowledge, this is the first effort that targeted on reducing the total tracking time required to track even capture a target in surveillance applications.

In the future endeavors, we aim to further work in different aspects. One possible aspect is to investigate more practical issues in using the planarization concept for tracking, especially in the case of a large amount of localization errors. Another possible aspect is to verify the proposed scheme in different tracking situations. For instance, intruders (e.g., people) are moving toward the fence of a surveillance region and are taking different directions. In this instance, it can be interesting to consider the edge crossing between the monitor and backup as the fence crossing of the surveillance region. One more aspect is to analyze the real-time issue and the cost of fault tolerance in both sparse and dense networks with complex scenarios.

Appendix A

Mitigating the Difficulties in Face Generation and Regeneration

In Sects. 3.1, 3.3.1, and 4.2.1, we have described the face generation at the system initialization and node organization into faces (or face regeneration) during tracking operation, respectively. In this appendix, we discuss some difficulties in face generation during target tracking.

According to the geographic routing algorithms, a node carries out routing through the boundaries of faces intersected by virtual edges, transmitting a packet from a source node u to a destination node v by employing a local *right-hand rule* technique. This is the technique by which a packet is forwarded along the next edge clockwise, or counterclockwise, from the edge where it arrived. Such a technique requires all faces to traverse completely on the way to a destination (more specifically, there exists no path from node u to node v) or to terminate when node v is eventually reached. This algorithm has limitations—they involve a number of faces, and require exchanging information between them. Thus, a large number of nodes of those faces are actively involved in carrying out an application task that may not be efficient for an envisaged WSN application. For example, if we considered all of the faces and the nodes that correspond to a particular node (e.g., the monitor), the frequency of interactions in the WSN during each tracking of target tracking would be high, resulting in a high energy consumption in the WSN.

Also in the planarized network model, the greedy forwarding technique work well in dense WSNs. But It experiences routing failures in more realistic, non-uniform node placements, where it suffers from so-called local minima, i.e., nodes with no direct neighbor closer to the destination than themselves [64, 66].

To make the techniques practicable in this application, we make some modifications on it. We determine the boundary of a face region rather than to generating only planar subgraphs. That can reduce the chance of having local minima [66].

We determine the boundary of a face region rather than to only generating a planar subgraph. In more clearly, we can determine a boundary/perimeter of a subset of nodes, that encloses all nodes of a connected component and contains all local minimum nodes on its border. This has to be achieved by a fully reactive and localized algorithm, which cannot rely on knowledge of the 1-hop neighborhood in advance. Locations of 1-hop neighbors have to be requested explicitly by message exchange. That can reduce the chance of having local minima.

We consider that a node u communicates to the nodes in the face to which it belongs and also maintains connectivity through its adjacent neighbors along the boundaries of the faces using the right-hand rule technique. Only the face, to which a node u belongs, is traversed completely and one of its immediately neighboring nodes, v, of the face is the destination node. More explicitly, a node u needs to know only its own, and its adjacent neighboring node v's locations. u grows its radio range R_c

$$\forall u \in V \quad r_c(u) = \max\{d(u, v) | v \in V \wedge (u, v) \in E_{RNG}\} \tag{22}$$

as in (22) until all its neighboring nodes are found in the not-yet-covered face. Thus, the number of faces which a node belongs to reduces to a few faces.

There is a class of protocols and algorithms in the literature focusing on network planarization that helps us to make the improvement on the correctness on face generation at the system initialization. These include disconnection between the nodes, incorrect edge removal, insufficient edge removals, cross-links, link or node failure, establishing direct link, hole/obstacle problem, non-ideal radio ranges, network instability, and so on [6, 51, 52, 64, 65, 74–77, 82–92]. Furthermore, we think that there may still remain some of the difficulties when using faces in a practical environment. This is why, we ensure to the preservation of faces by regenerating them at a certain moment; e.g., when the nodes in a face can detect t or receive a request to detect t before t's moving to the face. The face prediction helps to achieve this. We consider to maintain a linear link list between nodes of faces, which can help mitigate irregularities found in faces and can provide online fault tolerance.

In the following, we discuss some concerns/questions that are essential to make more clear the tracking through faces.

Concern 3 *Given the initial radio broadcasts on the generation of a planar graph, what if v_1 can also hear v_{12} and v_{20} in Fig. 2 during a tracking operation?*

Our response to this concern goes to the sensor node organization rules described in Sect. 4.1. Referring to Fig. 2, v_1 may hear both v_{12} and v_{20}. However, neither v_{12} nor v_{20} can be an adjacent neighboring node of v_1. To generate faces effectively, we put some preferences on nodes' interactions, by which both v_2 and v_5 cannot be the witness between v_1 and v_{12}, and between v_1 and v_{20}, respectively. If there are several neighbors reachable under the same radio range (e.g., v_1 may hear v_{12}, v_{20}, or may also hear v_{11}), a decision is made on this situation, which node should be the adjacent neighboring nodes of v_1; in other words, whether a node like v_{12} should be included in a face or not. There are two preferences to decide on the situation: (i) the minimal Euclidean distance from u to v; (ii) whether or not v is the first node by using the right-hand rule technique (having no witness). If these two conditions are not met, v_{12} is excluded from v_1's adjacent neighboring node list (i.e., connected to another face). Similarly, v_{20} is excluded from v_1's adjacent neighboring node list. We can emphatically say that they are neither the immediate neighbors nor the distant neighbors of v_1.

Concern 4 *According to the rules of node organization techniques described in Sect. 4.1, do all the nodes of all the neighboring faces of a specific face (where t is currently moving) involve in tracking?*

According to the rules of node organization into faces, we only consider a t's current moving face (F_i) and the predicted face F_j, and the edges and nodes of these faces associated with the monitor and backup nodes. We need to mention an underlying issue of the localized face: When t travels across F_i, not only is the closed face covered, but an additional area is also covered by each node's working area, since each node of F_i contributes a part of its working area inside F_i. The rest of the working area of the node is on the outside of F_i, which is included with a neighboring face

and is not mainly our concern, except that there is the event of t's missing. The outside/additional area of F_i can be calculated by the rest of the working areas of all of the nodes of F_i. In this example, the number of nodes of F_i is still the same. Such an area can be effectively used in the event of t's missing. If any node corresponding to F_i can detect t outside of F_i in its working area, the adjacent face of F_i immediately becomes the new F_i, and one of the node become monitor based on the detection probability (P_d) on t and another node becomes the backup, just like the target detection and tracking operation start at the beginning. They send a message to the monitoring and neighboring nodes in the previous F_i to change their active state s_i to inactive state s_2 (excluding the node if any node belongs to the new F_i). The calculation of additional area coverage can be found in existing localization approaches, e.g., [93].

Concern 5 *What is the main purpose to treat two nodes as "monitor" and "backup" for the tracking?*

In many localization techniques, such as TOA, TDOA, AOA, or MPS, a large number of nodes (3 to many) requires to participate in tracking. In many cases, most of the nodes of the network should be awake during a tracking. Tracking schemes with target localization-free require more nodes, such as *Forms* needs a lot of nodes (12 to many) in each period of tracking. In `t-Tracking`, one of the main purposes or advantages of treating the monitor and backup as the common nodes between the faces, F_i and F_j, is to minimize the number of participating nodes in target tracking. For example, v_5 has 5 adjacent faces with 12 neighboring nodes in Fig. 2, and node v_1 has 3 adjacent faces with 8 neighboring nodes. However, nodes (v_1 and v_5) together directly correspond only 2 faces with 7 neighbors. Thus, in a normal case of tracking, the number of nodes involved in tracking can be minimized from 20 to 7. This also implies that a number of nodes that hear about t's approaching become active. Among them, the number of nodes that do not participate in tracking (they may be basically idle and unnecessarily consume energy) can be minimized. Another purpose is to provide robustness in tracking. In the case that the monitor has any problem due to any reason, the backup takes the role of the monitor. Target detection and localization is mainly performed by using their previous "cooperation information".

Appendix B

More Concerns with the Proposed Tracking Algorithms

Concern 6 *Does P_d^c (the combined detection probability) estimation really result in a successful tracking operation?*

As t moves toward the monitor and backup, the distance $d(l_i^s - l_i)$ between t and each of the nodes decreases and the P_d^c between the monitor and backup increases. In most cases, P_d^c between the pairs of neighboring nodes should decrease. The steady

decrease in the distance increases in the probability, meaning that t is moving from the monitor and backup to new monitor and backup (from F_i to F_j, as shown in Fig. 3(b)). When P_d^c by any two nodes (the monitor and backup) is the highest, compared to its probabilities in the previous instants and to probabilities of the neighboring pairs of nodes, the chance that t crosses the edge between them is high, resulting normally in a successful tracking step (STS). We think that there may not happen such a case that P_d^c of two pairs of nodes is the same. However, because of the node organization into faces, there has a possibility that P_d^c of two pairs of nodes can be close.

Recall that in Sect. 4.2, we have mentioned that the angle between two or three adjacent neighboring nodes can be estimated to decide in which face t is in. This can also be used to reduce the above possibilities. However, such estimation may bring additional complexity in computation. Regarding the relevant information of the node organization into faces, we set three conditions for any pair of nodes to be the monitor and backup: (i) P_d^c should be higher than other pairs of nodes; (ii) they should be the adjacent and also be the immediate neighbors; (iii) they should be in the same face F_i (e.g., F_2).

These conditions reduce the wrong selection of the monitor and backup as well as the wrong detection of t, resulting in a successful tracking operation.

Concern 7 *Is there any big deal of detection error in the estimation of P_d^c between two nodes? More precisely, according to the scenario presented in Fig. 4, how does the system detect that t is going to cross between v_1 and v_5, not between v_1 and v_3, or between v_5 and v_4.*

This is quite similar to Concern 6. But we want to discuss it in another point of view, which may help understanding the target tracking through faces.

If t moves, it must move toward another face F_j, but initially the system does not know which face t is moving toward. In fact, tracking t is not only based on t's movement direction, also based on the direction of the nodes (or which face) t is moving toward. As shown in the scenario presented in Fig. 4a–c, it may move toward any face, i.e., toward F_3 by crossing the edge between v_6 and v_7, toward F_{12} by crossing the edge between v_5 and v_6, toward F_{10} by crossing the edge between v_7 and v_2, and F_{18} by crossing the edge between v_2 and v_1. It may also turn back to F_1 by crossing the edge between v_1 and v_5. In an extreme case, if t turns back, it should be easily detected as it is still under detection (within R_s of the same nodes). When such a case happens, $d(l_i^s - l_i)$ continues to decrease as time goes on, which was increasing in the previous some time instants. P_d^c between any pair of nodes should be increased or decreased (see Concern 6). This indicates that t is still moving inside F_i and may move across from F_i to F_j. There is no chance that t is crossing another edge between another pairs of nodes (e.g., between v_1 and v_3, between v_3 and v_4, between v_5 and v_{24}, or others) without going through the edge between the monitor and backup or the edge between any pair of nodes of F_i (which are also common nodes between F_i and F_j). Thus, there should be no big deal of detection error.

Concern 8 *Suppose that t is moving toward face F_7, how does it work? More precisely, how does the system knows that t is moving toward F_7, not F_3, or F_9 in Fig. 2? Actually, in the beginning, how does the system detect t is in F_1?*

To reduce the confusion about the target tracking, we raise Concern 7. In fact, this is similar to Concern 6 in the sense that we discuss Concern 6 considering t's moving through an edge between any two nodes (the monitor and backup), while we can discuss Concern 7 considering t's moving through an edge between two faces. The two nodes (a pair) are the common nodes between the two faces. In other words, the edge between the two nodes (a pair) is also the common edge between the two faces. The underlying techniques are the same. We emphatically say that, in `t-Tracking`, it is non-trivial to determine exactly at which face t is currently located and moving, and then which face t may move. We have elaborated how the above conditions can be obtained in Sect. 4.2 and Algorithm 2.

Concern 9 *Why is the "face prediction" preferred to the target movement prediction?*

Depending on finding accurate movement prediction, providing quality of tracking (QoT) is difficult in a practical, unpredictable environment. Because of faults in the WSN, noise in the environments, or any other reasons, if there are prediction errors, it becomes tough to relocate t further. Particularly, it is tough when considering the fast tracking operation and radio and energy constraints in the WSN. This is why, we first roughly find the movement information of t inside a face and the prediction information. Then, we convert these into a "face prediction". We do not find any second-order error adjustment on the localization or on the estimation of the P_d. Up to this point, we have assumed that t can be detected by the monitor and backup, which are the end nodes of a "direct edge" (the "common nodes" between two faces), that t is about to cross to. Thus, to avoid an unexpected loss of tracking, the tracking is based on the face prediction.

Concern 10 *What if t is no longer in the detection range of a F_i? We handle this extreme case that may appear during tracking operation. Assuming that t has disappeared or is missed during tracking, t may be out of R_s of any of monitor and backup. This may be due to several reasons: irregular signal patterns, especially when (i) a face size is too small in the situation that network density ρ too high or low in the local areas and (ii) t is much faster than normal. In such a case, the WSN in `t-Tracking` still has the ability to track/capture t. This case is mitigated by extending the face area coverage (see Concern 2).*

Suppose that there is the event of t's missing reported by the monitor for whatever reason. When the monitor broadcasts a message about the event, the distant nodes start sensing (or which are already in sensing function), since they are in s_0. If t is sensed by a distant node in its R_s, but is outside of F_i, the corresponding face in which t is sensed becomes a new F_i. The distant node becomes the monitor, and it issues a message to the nodes in the previous F_i and finds a backup. All of the nodes, including previous monitor and backup in the previous F_i, revert to S_2, and the nodes in the current F_i start target detection and face detection algorithm.

References

1. Naderan, M., Dehghan, M., Pedram, H., Hakami, V.: Survey of mobile object tracking protocols in wireless sensor networks: a network-centric perspective. Int. J. Ad Hoc Ubiquitous Comput. **11**(1), 34–63 (2012)
2. Wang, G., Bhuiyan, M.Z.A., Cao, J., Wu, J.: Detecting movements of a target using face tracking in wireless sensor networks. IEEE Trans. Parallel Distrib. Syst. **25**(4), 939–949 (2014)
3. Bhuiyan, M.Z.A., Wang, G., Cao, J., Wu, J.: Deploying wireless sensor networks with fault-tolerance for structural health monitoring. IEEE Trans. Comput. **64**(2), 382–395 (2015)
4. Bhuiyan, M.Z.A., Wang, G., Vasilakos, A.V.: Local area prediction-based mobile target tracking in wireless sensor networks. IEEE Trans. Comput. **64**(2), 1968–1982 (2015)
5. Bhuiyan, M.Z.A., Wang, J.W.G., Wang, T., Hassan, M.: e-Sampling: event-sensitive autonomous adaptive sensing and low-cost monitoring in networked sensing systems. ACM Trans. Auton. Adapt. Syst. 1–25 (2017)
6. Cartigny, J., Ingelrest, F., Simplot, D., Stojmenovic, I.: Localized LMST and RNG based minimum-energy broadcast protocols in Ad Hoc networks. Ad hoc Netw. (Elsevier) **3**(2), 1–16 (2005)
7. Bhuiyan, M.Z.A., Wang, J.W.G., Wang, T., Hassan, M.: Quality guaranteed and event-sensitive data collection and monitoring in wireless vibration sensor networks. IEEE Trans. Ind. Inform. 1–12 (2017). https://doi.org/10.1109/TII.2017.2665463
8. Yu, Y.: Consensus-based distributed linear filter for target tracking with uncertain noise statistics. IEEE Sens. J. **17**(15), 34–63, 12 (2017)
9. Yu, Y.: Wireless communications and mobile computing. IEEE Sens. J. **2017**(2017), 1–19, 12 (2017)
10. Amaldi, E., Capone, A., Cesana, M., Filippini, I.: Design of wireless sensor networks for mobile target detection. IEEE/ACM Trans. Netw. **20**(3), 784–797 (2012)
11. Song, L., Hatzinakos, D.: A cross-layer architecture of wireless sensor networks for target tracking. IEEE/ACM Trans. Netw. **15**(1), 145–159 (2007)
12. Keung, G.Y.: BoLi, Zhang, Q.: The intrusion detection in mobile sensor network. IEEE/ACM Trans. Netw. **20**(4), 1152–1161 (2013)
13. Sarkar, R., Gao, J.: Differential forms for target tracking and aggregate queries in distributed networks. IEEE/ACM Trans. Netw. **21**(4), 1159–1172 (2013). https://doi.org/10.1109/TNET.2012.2220857
14. Vicaire, P., He, T., Cao, Q., Yan, T., Zhou, G., Gu, L., Luo, L., Stoleru, R., Stankovic, J.A., Abdelzaher, T.F.: Achieving long-term surveillance in VigilNet. ACM Trans. Sens. Netw. **5**(1), 1–39 (2009)
15. He, T., Vicaire, P.A., Yan, T., Luo, L., Gu, L., Zhou, G., Stoleru, R., Cao, Q., Stankovic, J.A., Abdelzaher, T.F.: Achieving real-time target tracking using wireless sensor networks. In: Proceedings of the IEEE 12th Real-Time and Embedded Technology and Applications Symposium (RTAS'06), pp. 37–48 (2006)
16. Lin, C.-Y., Peng, W.-C., Tseng, Y.-C.: Efficient in-network moving object tracking in wireless sensor networks. IEEE Trans. Mob. Comput. **5**(8), 1044–1056 (2006)
17. Zhu, H., Li, M., Zhu, Y., Lionel, M.N.: HERO: online real-time vehicle tracking. IEEE Trans. Parallel Distrib. Syst. **20**(5), 740–752 (2009)
18. Bhuiyan, M.Z.A., Wu, J., Wang, G., Cao, J.: Sensing and decision-making in cyber-physical systems: the case of structural health monitoring. IEEE Trans. Ind. Inform. **12**(6), 2103–2114 (2016)
19. Bhuiyan, M.Z.A., Wang, G., Wu, J., Xiaofei, X., Liu, X.: Application-oriented sensor network architecture for dependable structural health monitoring. In: Proceedings of IEEE PRDC, pp. 134–147 (2015)
20. Bhuiyan, M.Z.A., Wang, G., Cao, J., Wu, J.: Energy and bandwidth-efficient wireless sensor networks for monitoring high-frequency events. In: Proceedings of the 10th IEEE International Conference on Sensing, Communication, and Networking (SECON'13), pp. 194–202 (2013)

21. Bhuiyan, M.Z.A., Wang, G., Wu, J., Cao, J., Liu, X., Wang, T.: Dependable structural helath monitoring using wireless sensor networks. IEEE Trans. Dependable Secure Comput. **14**(4), 363–376 (2017)
22. American Border Patrol, Operation B.E.E.F. Border enforcement evaluation first, 2006 [Online]. http://www.americanpatrol.com/ABP/BEEF/PDF/BEEFPLANDEC118.pdf
23. Luo, H., Wu, K., Guo, Z., Gu, L., Ni, L.M.: Ship detection with wireless sensor networks. IEEE Trans. Parallel Distrib. Syst. **23**(7), 1336–1343 (2012). http://doi.ieeecomputersociety.org/10.1109/TMC.2013.34
24. Jeong, J., Gu, Y., He, T., Du, D.H.C.: Virtual scanning algorithm for road network surveillance. IEEE Trans. Parallel Distrib. Syst. **21**(12), 1734–1749 (2010)
25. Bisdikian, C., Kaplan, L.M., Srivastava, M.B.: On the quality and value of information in sensor networks. ACM Trans. Sens. Netw. **9**(4), 1–26 (2013)
26. Chen, P., Zhong, Z., He, T.: Bubble trace: mobile target tracking under insufficient anchor coverage. In: Proceedings of the IEEE 31st International Conference on Distributed Computing Systems (ICDCS 2011, pp. 770–779 (2011)
27. Yedavalli, K., Krishnamachari, B.: Sequence-based localization in wireless sensor networks. ACM Trans. Mob. Comput. **7**(1), 81–84 (2005)
28. Ding, M., Cheng, X.: Fault tolerant target tracking in sensor networks. In: Proceedings of the ACM 10th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'09), pp. 125–134 (2009)
29. Bhuiyan, M.Z.A., Wang, G., Cao, J., Wu, J., Liu, X.: Localized decision making in wireless sensor networks for online structural health monitoring. Technical Report, Central South University, Changsha, China (2013). http://trust.csu.edu.cn/faculty/~csgjwang/publications/technicalreport/TR-SISE-02-Decision.pdf
30. Wang, X., Ma, J., Wang, S., Bi, D.: Distributed energy optimization for target tracking in wireless sensor networks. IEEE Trans. Mob. Comput. **9**(1), 73–86 (2010)
31. Misra, S., Singh, S.: Localized policy-based target tracking using wireless sensor networks. ACM Trans. Sens. Netw. **8**(3), 1–30 (2012)
32. Souza, E.L., Campos, A., Nakamura, E.F.: Tracking targets in quantized areas with wireless sensor networks. In: Procedings of IEEE 36th Conference on Local Computer Networks (LCN'11), pp. 235–238 (2011)
33. Bhuiyan, M.Z.A., Kuo, S.-Y., Wu, J.: Special issue on dependability in parallel and distributed systems and applications. Inf. Sci. **319**(1), 1–2 (2017)
34. Basheer, M., Jagannathan, S.: Localization and tracking of objects using cross-correlation of shadow fading noise. IEEE Trans. Mob. Comput. (IEEE Computer Society Digital Library) (2013). http://doi.ieeecomputersociety.org/10.1109/TMC.2013.34
35. Xu, E., Ding, Z., Dasgupta, S.: Target tracking and mobile sensor navigation in wireless sensor networks. IEEE Trans. Mob. Comput. **12**(1), 177–186 (2013)
36. Wang, X., Fu, M., Zhang, H.: Target tracking in wireless sensor networks based on the combination of KF and MLE using distance measurements. IEEE Trans. Mob. Comput. **11**(4), 567–576 (2012)
37. Tan, R., Xing, G., Liu, B., Wang, J., Jia, X.: Exploiting data fusion to improve the coverage of wireless sensor networks. IEEE/ACM Trans. Netw. **20**(2), 450–462 (2012)
38. Yang, Z., Jian, L., Wu, C., Liu, Y.: Beyond triangle inequality: sifting noisy and outlier distance measurements for localization. ACM Trans. Sens. Netw. **9**(2), 1–20 (2013)
39. Au, A.W.S., Feng, C., Valaee, S., Reyes, S., Sorour, S., Markowitz, S.N., Gold, D., Gordon, K., Eizenman, M.: Indoor tracking and navigation using received signal strength and compressive sensing on a mobile device. IEEE Trans. Mob. Comput. **12**(10), 2050–2062 (2013)
40. Klein, D.J., Venkateswaran, S., Isaacs, J.T., Burman, J., Pham, T., Hespanha, J., Madhow, U.: Localization with sparse acoustic sensor networks using uavs as information-seeking data mules. ACM Trans. Sens. Netw. **9**(3), 1–29 (2013)
41. Ercan, A.O., Gama, A., Guibas, L.J.: Object tracking in the presence of occlusions using multiple cameras: a sensor network approach. ACM Trans. Sens. Netw. **9**(2), 1–36 (2013)

42. Kwon, Y., Mechitov, K., Sundresh, S., Kim, W., Agha, G.: Resilient localization for sensor networks in outdoor environments. ACM Trans. Sens. Netw. **7**(1), 1–30 (2010)
43. Karbasi, A., Oh, S.: Robust localization from incomplete local information. IEEE/ACM Trans. Netw. **21**(4), 1131–1144 (2013)
44. Xiao, Q., Bu, K., Wang, Z., Xiao, B.: Robust localization against outliers in wireless sensor networks. ACM Trans. Sens. Netw. **9**(2), 1–26 (2013)
45. Chiu, W.-Y., Chen, B.-S., Yang, C.-Y.: Robust relative location estimation in wireless sensor networks with inexact position problems. IEEE Trans. Mob. Comput. **11**(6), 935–946 (2012)
46. De, D., Song, W.-Z., Xu, M., Wang, C.-L., Cook, D., Huo, X.: FindingHuMo: real-time tracking of motion trajectories from anonymous binary sensing in smart environments. In: Proceedings of the 2012 IEEE 32nd International Conference on Distributed Computing Systems (ICDCS '12), pp. 162–172 (2012)
47. Zheng, J., Bhuiyan, M.Z.A., Liang, S., Xing, X., Wang, G.: Auction-based adaptive sensor activation algorithm for target tracking in wireless sensor networks. Future Gener. Comput. Syst. **39**, 88–99 (2014)
48. Chen, F., Dressler, F.: A simulation model of IEEE 802.15.4 in OMNeT++. In: Proceedings of GI/ITG KuVS Fachgesprch Drahtlose Sensornetze, pp. 35–38 (2007)
49. Luo, J., Hubaux, J.-P.: Joint sink mobility and routing to increase the lifetime of wireless sensor networks: The case of constrained mobility. IEEE/ACM Trans. Netw. **18**(3), 871–884 (2010)
50. Li, X., Yang, J., Nayak, A., Stojmenovic, I.: Localized geographic routing to a mobile sink with guaranteed delivery in sensor networks. IEEE J. Sel. Areas Commun. **30**(9), 1719–1729 (2012)
51. Huang, Q., Bhattacharya, S., Lu, C., Roman, G.-C.: FAR: Face-aware routing for mobicast in large-scale sensor networks. ACM Trans. Sens. Netw. **1**(2), 240–271 (2005)
52. Leong, B., Mitra, S., Liskov, B.: Path vector face routing: geographic routing with local face information. In: Proceedings of the IEEE 13th International Conference on Network Protocols (ICNP'05), pp. 147–158 (2005)
53. Bhuiyan, M.Z.A., Wang, G., Wu, J.: Target tracking with monitor and backup sensors in wireless sensor networks. In: Proceedings of the IEEE 18th International Conference on Computer Communications and Networks (ICCCN'09), pp. 1–6 (2009)
54. Bhuiyan, M.Z.A., Wang, G., Wu, J.: Polygon-based tracking framework in surveillance wireless sensor networks. In: Proceedings of the IEEE 15th International Conference on Parallel and Distributed Systems (ICPADS'09), pp. 174–181 (2009)
55. Sathyan, T., Hedley, M.: Fast and accurate cooperative tracking in wireless networks. IEEE Trans. Mob. Comput. **12**(9), 1801–1813 (2013)
56. Singh, J., Kumar, R., Madhow, U., Suri, S., Cagley, R.: Multiple-target tracking with binary proximity sensors. ACM Trans. Sens. Netw. **8**(1), 1–26 (2011)
57. Li, H., Barbosa, P.R., Chong, E.K.P., Hannig, J., Kulkarni, S.R.: Zero-error target tracking with limited communication. IEEE J. Sel. Areas Commun. **26**(4), 686–684 (2008)
58. Jiang, B., Ravindran, B., Cho, H.: Probability-based prediction and sleep scheduling for energy-efficient target tracking in sensor networks. IEEE Trans. Mob. Comput. **12**(4), 735–747 (2013)
59. Gao, P., Shi, W., Zhou, W., Li, H., Wang, X.: A location predicting method for indoor mobile target localization in wireless sensor networks. Int. J. Distrib. Sens. Netw. **2013**, 1–11 (2013)
60. Jing, T., Hichem, S., Cedric, R.: Prediction-based cluster management for target tracking in wireless sensor networks. Wirel. Commun. Mob. Comput. **12**(9), 797–812, 2012 [Online]. http://dx.doi.org/10.1002/wcm.1014
61. Arik, M., Akan, O.B.: Collaborative mobile target imaging in UWB wireless radar sensor networks. IEEE J. Sel. Areas Commun. **28**(6), 950–961 (2010)
62. Dong, D., Liao, X., Liu, Y., Li, X.-Y., Pang, Z.: Sharp thresholds for relative neighborhood graphs in wireless ad hoc networks. Algorithmica **60**(3), 593–608 (2011)
63. Toussaint, G.: The relative neighborhood graph of finite planarset. Pattern Recognit. **12**(4), 261–268 (1980)
64. Ruehrup, S., Stojmenovic, I.: Optimizing communication overhead while reducing path length in beaconless georouting with guaranteed delivery for wireless sensor networks. IEEE Trans. Comput. (2013). http://doi.ieeecomputersociety.org/10.1109/TC.2012.148

65. Kim, Y.-J., Govindan, R., Karp, B., Shenker, S.: Lazy cross-link removal for geographic routing. In: Proceedings of ACM SenSys (2006)
66. Rhrup, S., Stojmenovic, I.: A new traversal scheme for georouting and boundary detection in WSNs. University of Ottawa, Canada, TR-2010-05 (2010)
67. Guibas, L.J., Hershberger, J.: Optimal shortest path queries in a simple polygon. J. Comput. Syst. Sci. **39**(2), 126–152 (1989)
68. Sugihara, R., Gupt, R.K.: Optimal speed control of mobile node for data collection in sensor networks. In: Proceedings of IEEE INFOCOM (2009)
69. Gu, Y., Ji, Y., Li, J., Zhao, B.: ESWC: efficient scheduling for the mobile sink in wireless sensor networks with delay constraint. IEEE Trans. Parallel Distrib. Syst. **24**(7), 1810–1820 (2013)
70. Xu, X., Luo, J., Zhang, Q.: Delay tolerant event collection in sensor networks with mobile sink. In: Proceedings of IEEE 29th Joint Annual Conference of the IEEE Computer and Communications Societies (INFOCOM'10) (2010)
71. Sugihara, R., Gupta, R.K.: Optimal speed control of mobile node for data collection in sensor networks. IEEE Trans. Mob. Comput. **9**(1), 127–139 (2010)
72. Mourad, F., Chehade, H., Snoussi, H., Yalaoui, F., Amodeo, L., Richard, C.: Controlled mobility sensor networks for target tracking using ant colony optimization. IEEE Trans. Mob. Comput. **11**(8), 1261–1273 (2012)
73. Welsh, M., Mainland, G.: Programming sensor networks using abstract regions. In: Proceedings of the USENIX 1st Symposium on Networked Systems Design and Implementation (NSDI'04), pp. 29–42 (2004)
74. Mathews, E., Frey, H.: A localized planarization algorithm for realistic wireless networks. In: Proceedings of the IEEE 12th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM'11), pp. 1–11 (2011)
75. Dong, D., Liao, X., Liu, Y., Li, X.-Y., Pang, Z.: Fine-grained location-free planarization in wireless sensor networks. IEEE Trans. Mob. Comput. **12**(5), 971–983 (2013)
76. Zhang, F., Jiang, A., Chen, J.: On the planarization of wireless sensor networks. Algorithmica **60**(3), 593–608 (2011)
77. Bhuiyan, M.Z.A., Wang, G., Cao, J., Wu, J.: Local monitoring and maintenance for operational wireless sensor networks. In: Proceedings the IEEE 11th International Symposium on Parallel and Distributed Processing with Applications (ISPA'13) (2013)
78. Lu, G., Sadagopan, N., Krishnamachari, B., Goe, A.: Delay efficient sleep scheduling in wireless sensor networks. In: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'06), pp. 2470–2481 (2005)
79. Fuemmeler, J.A., Veeravalli, V.V.: Smart sleeping policies for energy efficient tracking in sensor networks. IEEE Trans. Signal Process. **56**(5), 2091–2101 (2008)
80. http://docs.tinyos.net/index.php/Imote2
81. I.D.O.A.: http://docs.tinyos.net/tinywiki/index.php/Imote2
82. Huang, Q., Lu, C., Roman, G.-C.: Mobicast: Just-in-time multicast for sensor networks under spatiotemporal constraints. In: Proceedings of ACM/IEEE IPSN, pp. 442–457 (2003)
83. Kim, Y.-J., Govindan, R., Karp, B., Shenker, S.: Geographic routing made practical. In: Proceedings of the USENIX 1st Symposium on Networked Systems Design and Implementation (NSDI'05), pp. 217–230 (2005)
84. Zhang, F., Jiang, A., Chen, J.: Robust planarization of unlocalized wireless sensor networks. In: Proceedings of IEEE INFOCOM, pp. 798–806 (2008)
85. Seada, K., Zuniga, M., Helmy, A., Krishnamachari, B.: Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. In: Proceedings of ACM SenSys, pp. 108–121 (2004)
86. Seada, K., Helmy, A., Govindan, R.: Modeling and analyzing the correctness of geographic face routing under realistic conditions. Ad Hoc Netw. **5**(6), 855–871 (2007)
87. Lin, J., Kuo, G.-S.: A novel location-fault-tolerant geographic routing scheme for wireless ad hoc networks. In: Proceedings of IEEE VTC, pp. 1092–1096 (2006)
88. Delouille, V., Neelamani, R., Baraniuk, R.: Robust distributed estimation in sensor networks using the embedded polygons algorithm. In: Proceedings of the ACM 3rd Information Processing in Sensor Networks (IPSN'04), pp. 405–413 (2004)

89. Frey, H., Stojmenovic, I.: On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks. In: Proceedings of ACM MobiCom, pp. 390–401 (2006)
90. Clouser, T., Miyashita, M., Nesterenko, M.: Fast geometric routing with concurrent face traversal. In: Proceedings of OPODIS, pp. 346–362 (2008)
91. Rao, A., Papadimitriou, C., Shenker, S., Stoica, I.: Geographical routing without location information. In: Proceedings of ACM MobiCom, pp. 96–108 (2003)
92. Lin, C.-H., Liu, B.-H., Yang, H.-Y., Kao, C.-Y., Tsai, M.-J.: Virtual-coordinate-based delivery-guaranteed routing protocol in wireless sensor networks with unidirectional links. In: Proceedings of IEEE INFOCOM, pp. 351–359 (2008)
93. Waelchli, M., Scheidegger, M., Braun, T.: Intensity-based event localization in wireless sensor networks. In: Proceedings of the IFIP 3rd Annual Conference on Wireless On demand Network Systems and Services (WONS'06), pp. 41–49 (2006)

Part V
Data Dissemination and Fusion

Data Dissemination and Remote Control in Wireless Sensor Networks

Xiaolong Zheng and Yuan He

Abstract Wireless Sensor Networks (WSNs) have been applied in a variety of application areas. Most WSN systems, once deployed, are expected to operate for a long period. Besides gathering data to the sink, reliable and efficient delivery of data or control messages from the sink to the sensors are an equally important task. During the lifetime, it is necessary to fix bugs, reconfigure system parameters, and upgrade the software to achieve reliable system performance. However, manually collecting all nodes back and reconfiguring the nodes through serial connections with computer is infeasible because it is labor-intensive and inconvenient. Data dissemination and remote control over a multi-hop network are desired to facilitate such tasks. In this book chapter, we are going to present the challenges and research space of data dissemination and remote control in WSNs, review existing approaches, introduce relevant techniques, assess various performance metrics, compare representative methodologies, and discuss the potential directions. We compare and elaborate on the existing approaches in two categories: structure-less approaches and structure-based approaches, classified by whether or not the network structure information is used during the disseminating process. Meanwhile, application of emerging techniques like concurrent transmissions in data dissemination further broadens the design space. We will also introduce the latest progress in those relevant directions.

1 Introduction

Wireless Sensor Networks (WSNs) have been attracting extensive attention from both the scientific community and industry [1, 2]. The development of low-cost, low-power, multi-functional sensor devices, boosting the application of WSNs. Real application systems are extensively deployed in a variety of application areas such as

X. Zheng · Y. He (✉)
School of Software, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing, China
e-mail: heyuan@mail.tsinghua.edu.cn

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_13

structural health protection [3, 4], environment monitoring [5–7], underground coal mine monitoring [8], event detection in targeted monitoring areas [9], and habitat monitoring [10].

To support aforementioned applications, remote control in WSNs is essential. Most of these WSN systems, once deployed, are intended to operate unattended for quite a long period. During their lifetimes, it is inevitable to fix software bugs [11, 12], remotely control system parameters [13, 14], and reprogramming the software [15, 16] in order to achieve reliable system performance. Unfortunately, for a large WSN system, manually collecting and reconfiguring nodes are infeasible because some WSN systems are deployed in areas where it is physically impossible for human beings to access [17]. It is also labor-intensive due to the huge number of nodes. Hence, both the system users and administrators require controlling WSNs remotely to manage the system services and system operation.

Disseminating data and control commands over a multi-hop network are a core building block of remote control in WSNs. As shown in Fig. 1, data dissemination represents delivering interested data (control commands, running parameters, new codes, etc.) from the sink node to all or selected nodes in the network, over multi-hop transmissions. Data dissemination is an inverse of data collection that generates data flows from sensing nodes to the sink node to mainly delivery sensing data, as shown in Fig. 2. Data dissemination and collection need to cooperate in the software-defined wireless networks to collect data and disseminate routing control messages [18–20].

In this book chapter, we discuss the challenges and the design requirements in data dissemination and remote control. We introduce existing methods by dividing them into two categories: structure-less schemes and structure-based schemes. Generally speaking, according to different assumptions about the knowledge of network structure information, existing data dissemination schemes can be classified into these two categories. In structure-based schemes, it is assumed that knowledge of network structure information such as node location or network topology is

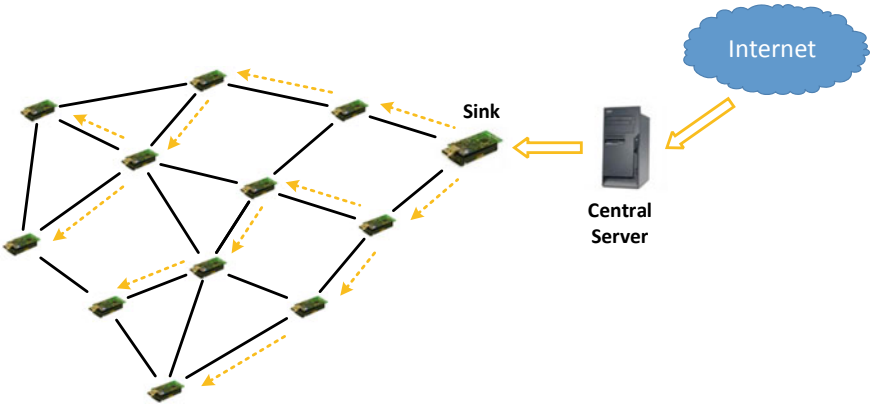


Fig. 1 Data dissemination in WSN

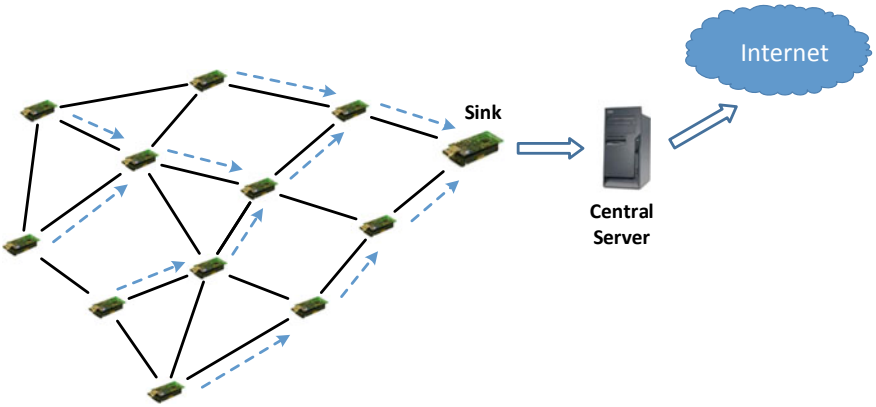


Fig. 2 Data collection in WSN

available. Dissemination methods can then take advantage of this knowledge to construct a dedicated structure for efficient dissemination. In structure-less schemes, there is no information of network structure or dedicated structure for dissemination.

For structure-less schemes, according to using negotiation mechanism or not, we further divide them into non-negotiation schemes and negotiation-based schemes. In negotiation mechanisms, the redundant transmissions are under control and the reliability is usually guaranteed. However, negotiation is a double-edged sword. It brings about additional communication overhead and time consumption as well. In non-negotiation schemes, without control message, the dissemination process is relatively quick but providing arbitrarily high reliability is nearly impossible. Besides, broadcast storm problem is easy to occur if non-negotiation schemes do not properly control the broadcast of nodes.

In structure-based schemes, two sub-categories, plain-structure and hierarchy-structure schemes, are introduced. In plain-based schemes, all nodes have the same role in the disseminating process. The structure information is only used for controlling the redundancy but not the forwarding strategy. On the other hand, in hierarchy-structure schemes, the structure information is used to construct a special structure dedicated for data disseminating. The network is usually divided into clusters with a cluster head for each. The cluster heads build up a backbone network to get data preferentially. And then the cluster headers disseminate data to the cluster members in their own clusters in parallel.

In this book chapter, we analyze the merits and demerits of corresponding schemes, along with the trade-offs between different categories. In the existing literature, different categories usually have definite boundaries and no trade-off has been analyzed before. The hybrid schemes, combining non-negotiation and negotiation-based schemes, will be discussed and analyzed in this chapter.

The rest of this chapter is organized as follows. In Sect. 2, we discuss the challenges and the requirements of data dissemination. Then we introduce the structure-less data dissemination scheme in Sect. 3 and structure-based data dissemination scheme in

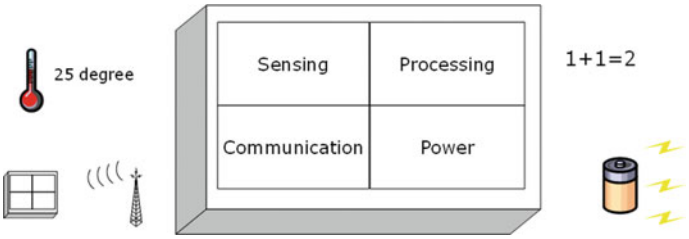


Fig. 3 Typical components of a WSN node

Sect. 4. In Sect. 5, we present some related techniques that can be employed in data dissemination. Then, we summarize performance measurement metrics and present the performance comparisons of existing data dissemination schemes in Sect. 6. We further discuss the open issues in Sect. 7 and give the conclusion in Sect. 8.

2 Requirements and Challenges

In this section, we first review the features of WSNs in Sect. 2.1. Then we analyze the data dissemination requirements in Sect. 2.2 and summarize the corresponding challenges satisfying the requirements in Sect. 2.3.

2.1 Features of WSNs

In WSN systems, it is common to deploy many sensor nodes in targeted areas which may be physically inaccessible for human beings. For example, for virgin forest monitoring application such as GreenOrbs [5], the deployed environment is not easy to access for system administrators to maintain the network on site.

A typical WSN node consists of four components: power, processing, sensing, and communication, as shown in Fig. 3. The power component is usually using battery when the target environment is outdoor without power supply around. Since the power is limited, other components have to be as low power as possible.

Due to the simple hardware of nodes and application requirements, compared to traditional wireless communication systems, WSNs have the following outstanding features.

- Large scale. Due to applications of WSN usually requiring large deployment area, the amount of sensor nodes can be very large because the communicating range is only dozens of meters.
- Long term. The WSN system is usually intended to work for a long time after deployment. A long lifetime is key to good applicability.

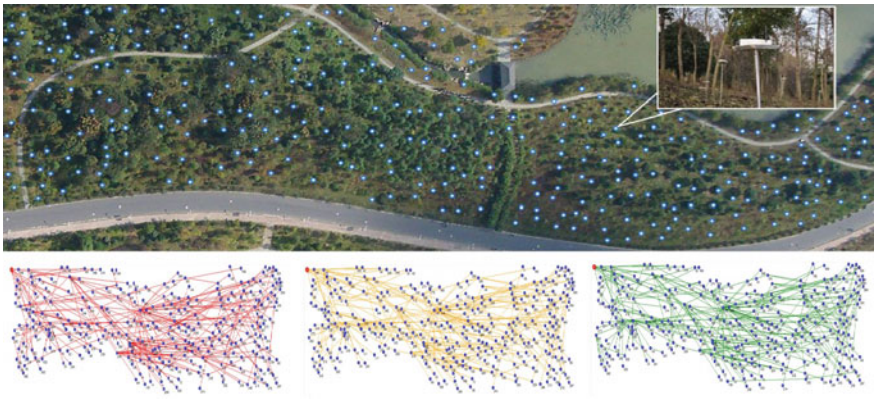


Fig. 4 Deployed prototype of GreenOrbs and the network topology

- Self-organized. WSNs are ad hoc networks which organize and manage the network by themselves. Hence, nodes usually adopt self-repairing and recovery mechanisms to be fault tolerant.
- Limited resources. As introduced above, sensor nodes are powered by battery, which means limited energy resources [21]. Besides, data storage and computational capacity are also limited because low-power hardware is adopted.
- Low power. In WSNs, sensor nodes usually work in a so-called duty-cycle mode [22–24] to fulfill the long-term lifetime requirement with limited energy resource. In duty-cycle mode, a node periodically wakes up for some tasks and then sleeps for most of the time. In sleeping mode, the consumed power is reduced to the minimum.
- Dynamic topology. WSNs are ad hoc networks which maintain the network by themselves according to the quality of communication links. But, communication links are easily affected by environmental changes. Therefore, WSNs are usually dynamic because the environment usually keeps changing, as shown in Fig. 4.

2.2 Requirements of Data Dissemination

As introduced in Fig. 1, data dissemination spreads data from sink node(s) to all nodes in network over a multi-hop network. Data can be a new code image for application changes, system commands, or updated system parameters for remote control. To satisfy the Quality of Services (QoS), three requirements of data dissemination in WSNs need to meet.

Reliability. Data dissemination generally requires 100% or near 100% reliability. Here 100% reliability has two meanings: (1) all the nodes in the network should get the data; (2) every node receives the whole data block without any hiatus. Reliability

is the most important and essential requirement for data dissemination. Because data dissemination is the building block of reprogramming and remote control, any mismatch of above two aspects can lead to inconsistency and even crash of the whole network.

Efficiency. Efficiency requirement has two aspects: time efficiency and energy efficiency. To achieve time efficiency, the dissemination process should finish as soon as possible. During the dissemination process, a large number of data dissemination packets occupy the channel. The normal functional collecting traffic is blocked, and the system is inefficacious during this period. Therefore, this interruption time period should be as short as possible.

Energy efficiency desires minimal energy consumption for the data dissemination process. This is the consequence of limited power resources, the feature of WSNs. The consumed energy consists of read–write of flash, radio transmission, and idle listening of radio. The read–write of flash is inevitable to store data blocks if the data block is large. The major of energy consumption comes from radioactivity which should be carefully controlled to achieve energy efficiency. Restricting the radio-on time during data dissemination is the key to energy efficiency. Energy balance between neighboring nodes can also alleviate the energy drain problem of nodes with large data traffic.

Scalability. WSN systems usually have various network size and density. A good data dissemination scheme should adjust to any scale, in terms of the number of nodes and node density. From the performance aspect, if the completion time of dissemination is linearly increasing with network scale, the dissemination scheme is regarded as scalable.

2.3 Challenges of Data Dissemination

Considering the distinctive features of WSNs, data dissemination faces several challenges.

Limited resources bring limitations to the dissemination process. Dissemination must adjust according to the capacity of node platforms. For example, if the disseminated data block are larger than the size of RAM on sensor node, it is impossible to transmit whole data block at one time. For example, Mica [25], released in 2001, has 4 KB RAM and 128 KB flash; Telos [26], released in 2004, has 10 KB RAM and 48 KB flash. If the size of data block is larger than the RAM size, transferring from RAM to flash is necessary.

As discussed in previous section, to be energy efficient, dissemination should try to minimize the radio-on time. The larger the data are, the more energy is consumed. Hence, except for the obligatory transmission cost of data packets, the radio for other extra use such as control messages and idle listening should be reduced as much as possible.

High reliability is always desired by dissemination but it is not trivial to achieve. As we know, the network is dynamic which means nodes in the network may break away from the network and new nodes can join in the network at any time. The connectivity of network keeps changing all the time. Data dissemination should deal with network dynamics and guarantee the new joined nodes also have the ability to catch up and get the latest data.

Due to the larger number of dissemination packets, it is easy to encounter the so-called broadcast storm problem [27]. Too many redundant transmissions result in serious packet collisions and transmission failures. To make data dissemination reliable and efficient, the broadcast storm problem should be avoided.

The simple reverse uses of routing protocol are not efficient. Routing protocols are designed for data collection. Data collection is usually a many-to-one communication model but data dissemination is one-to-many communication model. In data collection, the data flows are bottom-up because all nodes in the network send sensing data to a sink. But in data dissemination, all nodes receive new data from the sink. The generated data flows are top-down and one-to-many. General routing protocols fail to disseminate data efficiently. Dedicated dissemination scheme is necessary to accomplish high-reliable, high-efficient data dissemination. Many protocols tailored to data dissemination are proposed, as introduced in the following sections. Challenges of data dissemination in WSNs are also discussed in a survey [28].

3 Structure-Less Data Dissemination Schemes

We call the dissemination methods without dedicated dissemination structure as structure-less data dissemination schemes. The topology information of underlying network is not required for structure-less data dissemination schemes. As a compromise, local optimal strategy is adopted to greedily approximate global optimal solution. The major disadvantage of these schemes is that they are not able to achieve a global optimal solution. The advantage is that no additional overhead costed by getting global topology information and constructing a dedicated structure. Especially in the dynamic networks, frequent topology changes cause significant additional overhead. We further divide structure-less schemes into two categories, non-negotiation schemes and negotiation-based schemes, according to the use of negotiation strategy in a particular scheme.

3.1 Non-negotiation Schemes

The most representative non-negotiation scheme is *Classic flooding*. In classic flooding, the sink node starts the dissemination. Data is broadcasted by sink to all neighbors. Upon receiving a piece of data, the node will check the data are stored before or not. If not, the receiving node will store the data and then checks whether it has

already forwarded the data to its own neighbors. If not, it forwards a copy of the data to its neighbors by broadcast. Otherwise, it remains silent. Classic flooding maintains quite small amount of protocol state and disseminates data quickly in a network. It is simple yet effective.

However, the deficiencies of classic flooding lead to its inadequacy as a protocol for WSNs. First, classic flooding is unreliable. In this approach, there is no automatic repeat request (ARQ) scheme adopted and hence no reliability guarantees. ARQ is an error correction method for data transmission that uses acknowledgments (messages sent by the receiver indicating that it has correctly received a data frame or packet) and time-outs (specified periods of time allowed to elapse before an acknowledgment is to be received) to achieve reliable data transmission over an unreliable service. It is obvious that the fraction of receiving data is decreasing exponentially with the increase in the hop count. However, high reliability is a desired characteristic and a basic requirement. As a result, classic flooding has to repeatedly broadcast data many times to provide a high reliability, resulting in less efficiency.

Second, classic flooding is easy to cause the so-called broadcast storm problem [27]. In classic flooding, no matter the neighbors have already received all data or not, a node receiving new data always rebroadcasts the new data. A node may hear several senders in its transmission ranges due to the broadcast nature of wireless communication. Too many rebroadcasts then lead to redundancy and collision. What's worse, if ARQ is employed for the reliability, the broadcast storm problem is more serious because all receivers send the ACK back immediately after receiving the broadcast packets. Therefore, if not properly used, classic flooding will result in serious redundancy and collision. In [27], the impacts and severity of broadcast storm are analyzed. The results show that the broadcast storm problem is aggravated by high density and large scale.

To conquer the broadcast storm problem, the authors in [27] proposed five revised flooding schemes: probabilistic, counter-based, distance-based, location-based, and cluster-based flooding. Among these schemes, probabilistic and counter-based flooding are two lightweight schemes that only slightly modify classic flooding. Distance-based and location-based schemes need the geography information to reduce the redundancy. Cluster-based scheme is a structure-based scheme, which will be introduced in the following section.

All these revised schemes designed to alleviate the broadcast storm problem are using two ways: (1) reducing broadcast redundancy; (2) differentiating stagger the broadcast time. First of all, all these schemes differentiate the rebroadcast timing by inserting a random back-off before the rebroadcast. Because the radio following IEEE 802.15.4 is simple without techniques like rate adaption in WLANs [29], the proposed methods just drop the broadcast to reduce the interference. Different schemes follow different dropping rules.

Probabilistic scheme is a straightforward scheme that reduces the redundant rebroadcasts by assigning a rebroadcast probability. On receiving a fraction of data for the first time, a node will rebroadcast it with the probability P. Clearly, when $P = 1$, probabilistic scheme is equivalent to classic flooding. But when $P < 1$, probabilistic scheme can avoid unnecessary rebroadcasts by adjusting the rebroadcast probability.

Counter-based scheme is also a scheme that tries to reduce the redundancy by suppressing rebroadcast. It is a density-aware method that leverage the density to adjust the rebroadcast behavior. When a node has a new data packet to transmit in the queue, it will overhear the channel status. If more than C packets identical to the packet in queue are overheard during the waiting period, the node will drop its own transmission.

All the parameters such as P and C are preset empirically. When a dynamic network is considered, the fixed preset parameters will be inappropriate. To deal with dynamic network, the authors in [30] propose the revised versions of probabilistic and counter-based flooding schemes. Instead of using the fixed preset parameters as previous work, the proposed schemes allow nodes to choose parameters based on local information. Take the network with topology shown in Fig. 5 as an example. Node B and C should have different rebroadcasting probabilities due to the different local network features. According to [30], the rebroadcasting probability of C should be smaller than the rebroadcasting probability of B. This is because that C's local network and its rebroadcast may bring less benefit and more harm to the whole process.

Gossip [31] is proposed to alleviate the storm problem by randomization. Actually, it is also a probabilistic scheme similar to probabilistic flooding [27]. But Gossip is an adaptive gossip protocol.

A node with Gossip will decide its gossip probability based on the number of its neighbors. The probability of a node is in inverse proportion to the number of its neighbors. We regard this scheme as the *Adaptive Neighbor* method. In this scheme, it only considers the transmission efficiency by assigning the probabilities adaptively. However, it ignores the special characteristics of network topology, network connectivity. Again, take the network with topology shown in Fig. 5 as an example. Suppose node A is the sink node who initiates the data dissemination. Based on the network connectivity, we can easily find that node G can only get data from C. Therefore, to

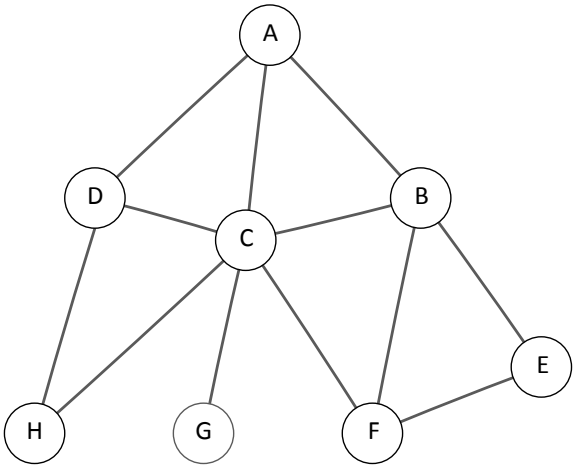


Fig. 5 An exemplified network topology

reliably disseminate data to whole network, node C must rebroadcast every packet to its child G. However, based on the strategy of *Adaptive Neighbor*, node C will choose a small rebroadcasting probability due to the large number of neighbors.

Trickle was proposed in [32] as a version management method that manages how and when the update of codes is performed. In Trickle, special data summaries that contain the version of data on the nodes are periodically broadcasted. The data summaries are used to maintain consistency in local areas. If a node receives a data summary with newer data version, it will request the latest data to keep itself consistent to other nodes with new data. Trickle adopts an exponential back-off timer to reduce the additional overhead caused by unnecessary summary exchanges. If a local network is consistent, then no data exchange is necessary. Then the interval between two data summaries can be enlarged to reduce overhead. Otherwise, the period is quite short to quickly discover new data and efficiently turn into data dissemination. Trickle adopts a suppression mechanism to reduce the unnecessary data summaries. A node will drop its own broadcast when an identical data summary is heard. Besides, *Adaptive Neighbor* scheme is inherited in this work. The rebroadcast probability is also adjusted based on the number of duplicated overhearing of the last message. Consider the topology in Fig. 5, again the deficiency is found. Because node C will overhear duplicate summaries from node A, B, and D, Node C will drop many broadcasts which will cause node G has less opportunities to get the new data.

Smart Gossip [33] takes underlying network topology into consideration to solve the above problem. Smart Gossip adjusts the rebroadcast probabilities according to the underlying network topology. The reason for the improper decisions in the above two schemes is that a node chooses its rebroadcasting probability independently. Hence, in Smart Gossip, the node dependency is defined and used for rebroadcasting probability adjustment. Smart Gossip defines node X "*depends on*" Y as the relationship between X and Y if Y gets data from X. A stronger dependency means the probability that Y is able to get data from other nodes except X is lower. In Fig. 5, we can find node G depends on node C totally. Hence, in Smart Gossip, C can learn the knowledge that G depends on it totally. After knowing the dependency, node C will increase the rebroadcasting probability up to 1. Despite of making adjustments based on dependency reflected by topology information, Smart Gossip is still divided into structure-less schemes because it only uses local topology which can be obtained in a decentralized manner.

3.2 Negotiation-Based Schemes

Negotiation is first proposed in *SPIN* [34, 35]. In negotiation methods, nodes in local areas negotiate with each other about who should be the forwarder and what is to be transmitted before the neighboring nodes really begin transmitting data. The so-called negotiation strategy is designed to conquer the broadcast storm problem and to obtain high reliability. Through negotiation about missing data, which part of the data is still needed to retransmit can be known be the senders. Then only useful

information will be transferred. Besides, the negotiation messages can act as NACKs that request the missing data. Hence, the reliability is guaranteed. Due to the reliability guarantee, it is more common use negotiation-based schemes to accomplish remote control than non-negotiation schemes. On the other hand, the negotiation about who should act as forwarder tries to keep one and only one node transmits the data in a local area. Thanks to the negotiation, the number of redundant data transmissions is dramatically reduced and the broadcast storm problem is avoided.

Generally speaking, negotiation uses three types of messages to do the dissemination through a three-handshake message exchange.

- ADV, advertisement messages. ADV messages specify the data that a sender wants to share in the form of meta-data.
- REQ, request for data. Nodes will send back requests to specify which packets are wanted, after they receiving the ADV.
- DATA, the requested data packets. The source node packs the requested packets and broadcasts it in the type of DATA.

Sensor Protocols for Information via Negotiation (*SPIN*) is a family of negotiation-based adaptive data dissemination protocols. The SPIN protocols share two common basic ideas. The first one is information sharing to avoid redundant transmissions. Nodes share the information about which parts of the data they already have and which parts are still needed. However, exchanging real data may be a costly network operation because the size of disseminated data are usually large. However, exchanging the summaries about disseminating data can reduce the cost. Meta-data is then proposed to succinctly and completely describe the latest data. Then the original large data can be identified by the meta-data with small size. It means the meta-data of the distinguishable data will still be distinguishable. On the other hand, the indistinguishable data will result in identical meta-data. Second, energy balancing is taken into consideration in *SPIN*. To prolong the lifetime of the system, the nodes monitor their own energy resources and adjust the dissemination behaviors correspondingly.

SPIN is the pioneer work that designs negotiation strategy in WSNs. The follow-up work usually adopts the similar negotiation strategy. Each node with *SPIN* broadcasts ADV messages to announce which data they have. If an ADV message is received, the receiver will check its own meta-data and compare it with the meta-data in the ADV message. If the meta-data are not matched, the receiver knows that the ADV sender possesses data with a newer version. The receiver then responds the sender by sending a REQ message that specifies which parts of the data is wanted by the receiver. Once receiving REQ messages, the sender will send out required data encapsulated in DATA messages.

Figure 6 shows an example of the negotiation procedure in *SPIN*. Node A initiates the dissemination. It broadcasts ADV messages to announce its possession of new data (1). Upon receiving ADV messages, node B, C, and D send out the REQ message which contains the information of wanted data parts (2). Node A combines all REQ messages and integrates the wanted data into next transmitted data block. And then, node A broadcasts DATA messages that contain all the data requested by the neighbors (3). Taking the lossy link into account, the DATA packets can get lose due

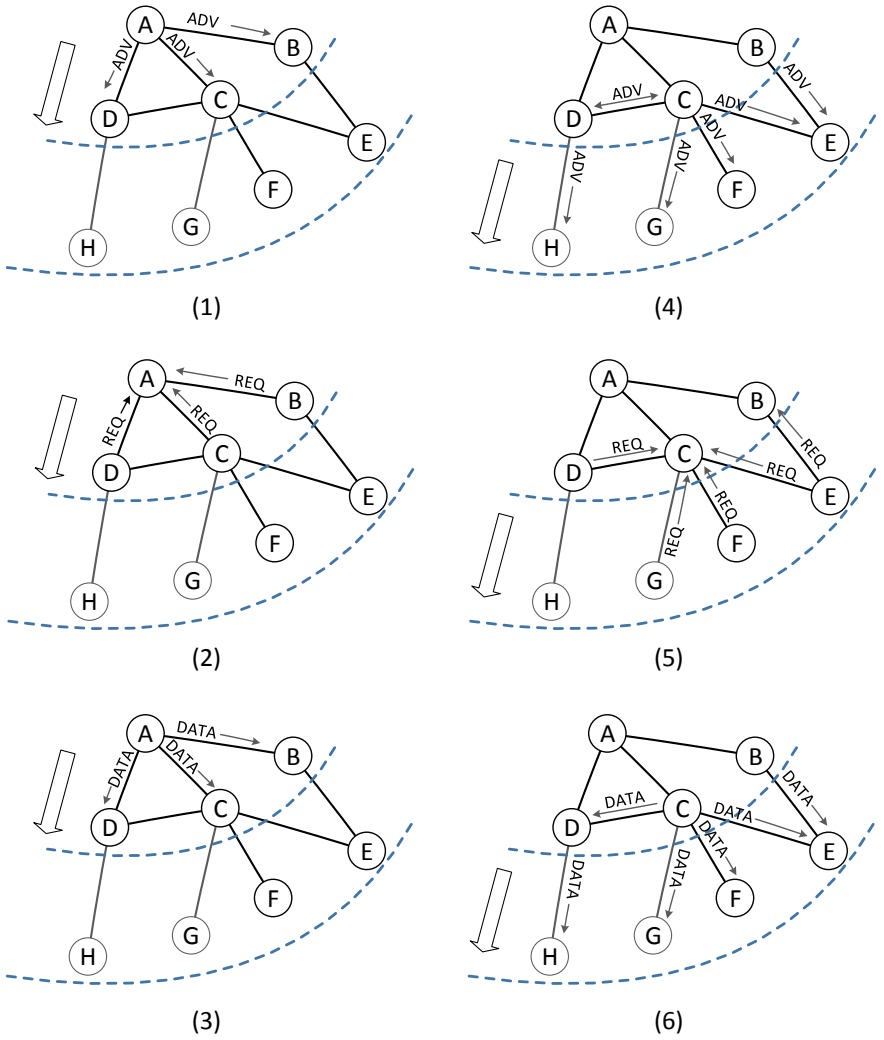


Fig. 6 Negotiation process in SPIN

to the poor link quality. The procedure (2) and procedure (3) are repeated until all the required DATA packets are received or reaching the maximum number of retries. If a node does not get the whole data after the maximum retries, it stops sending REQ anyway to avoid continuously using the poor links. It will wait for the next ADV message to start a new round negotiation and get what it needs from other senders. The nodes can act as forwarders after receiving all the requested data packets. They will broadcast ADV messages to announce their availability as forwarders (4). Then the neighbors of forwarders can request the data by sending REQ messages back to

the forwarders, e.g., node C, after hearing C's ADV message (5). Then by the way same to the one in (2) and (3), node C will broadcast the request data to its neighbors (6). By this hop-by-hop forwarding, data dissemination is accomplished.

In [36], the authors notice that multiple potential forwarders will have channel contention during the forwarding process. All the receivers that have new data can sever as forwarders, which one should be the forwarder in this local areas? If too many forwarders are broadcasting the data at the same time, packet collision between these forwarders will be serious. The dissemination efficiency will be hurt. To deal with above problem, the authors propose *Multi-hop Over-the-Air Programming (MOAP)*. MOAP selects only a small subset of the nodes in a neighborhood to act as forwarders, reducing the unnecessary traffic. Retransmissions are done through unicast to reduce the traffic further. The repair mechanism is a simple sliding window based retransmitting method.

Deluge [37] is a reliable data dissemination protocol built on *Trickle* (see Sect. 3.1) to accomplish version control. Similar to *SPIN*, Deluge also adopts a negotiation strategy. Deluge supports bulk data dissemination with a three-handshake negotiation strategy. Deluge can guarantee the reliability by the NACK messages. Data requester should specify the needed packets in the REQ messages. Then the forwarders only retransmit the missing packets to reduce unnecessary transmission cost. Deluge leverages the segmentation and pipelining mechanism which MOAP does not use. We will introduce segmentation and pipelining mechanism in Sect. 5.1. Due to these methods, Deluge is able to exploit the spatial multiplexing to speed up the dissemination process by concurrent transmission without collision. The reliability is guaranteed by hop-by-hop error recovery.

MNP [38] shares similar ideas with Deluge. MNP adopts the same negotiation framework as Deluge. The key difference is that a sender selection mechanism is proposed in MNP to reduce the probability of selecting inappropriate forwarder in Deluge. Recall that in Deluge, a candidate is randomly picked out among the neighbors as the forwarder, if there is more than one candidate senders in a local area. However, the number of forwarders will not be bounded and is still large enough to have the forwarder contention problem. What's more, if the randomly selected forwarder has poor link qualities to its neighbors, the completion time will be prolonged. MNP tries to solve this problem by proposing a greedy sender selection algorithm. In MNP, the candidate forwarder with largest expected *impact* will be greedily selected as the next forwarder in a local area. *Impact* is defined as the expected number of neighbors that can get data from a node if they select the node as the forwarder.

ECD [39] is an efficient code dissemination protocol that considers the unreliable wireless links. By leveraging one-hop neighbor link quality information, ECD tries to accurately measure the $impact$ of a sender. In MNP, $impact$ is used as the sender selection metric. However, the $impact$ of a node only considers the number of neighbors that need data from the node, without any consideration of the link quality. MNP can perform well in the networks where the links are reliable or near

reliable. But when the links are lossy, the measurement of $impact$ is not accurate. Different from MNP, ECD calculates the $impact$ with link quality as follows.

$$Impact(i) = \sum_{k \in N(i)} 1 \cdot p(i,k) \tag{1}$$

where $N(i)$ is the set of the uncovered neighbors of node i, $p(i,k)$ is the link quality from i to k. From Eq. (1), we can see the $impact$ of node i in ECD actually reflects the expected number of packets that the uncovered neighbors can receive if node i is selected as the forwarder.

ReMo [40] is a reliable reprogramming protocol. It is tailored to the mobile WSNs. In the mobile WSNs, the location of a sensor node is changing and uncertain. The mobility leads to a dynamic network condition such as varying link qualities. This intrinsic features of mobile WSNs bring challenges to the dissemination protocol design. First, the uncertain locations make it hard to get neighbor information for the potential senders. Second, sender selection algorithms do not work well because some information is not easy to get. For example, the Packet Reception Ratio (PRR) information in ECD is not available in mobile WSNs because the mobility causes continuous changes of the network condition.

To overcome above challenges, ReMo is proposed. Nodes with ReMo learn the neighbors' information such as link qualities and relative distances by measuring the link quality indicator (LQI) and received signal strength indicator (RSSI) of the received packets. Based on the measurement results, the nodes in local area select the best node for data exchanging, with the consideration of the movement paths. ReMo also adopts the polite gossip advertisements similar to *Trickle* (see Sect. 3.1). Broadcasting an advertisement or not is decided by the network density. Another distinguishable feature of ReMo is disordering reception of data. Remo allows disordered data and rearrange the order after all data are received because the mobility of nodes makes hop-by-hop reliability guarantee hard.

Besides the researches of traditional data dissemination, methods from other researcher areas can be leveraged to accomplish data dissemination. For example, publish/subscribe systems over WSNs [41, 42] can be used for data dissemination. in a publish/subscribe system, a subscriber sends subscribe messages which contain its interest to an event or a predefined topic, to the publisher. The publishers who receive the subscribe messages will send the publish messages according to the notification of subscribers' interests. More details and recent works can be found in the survey [43]. Such a message model is similar to the negotiation-based message model in data dissemination. In publish/subscribe systems, it is the subscribers (normal nodes in the network) initiating the process to pull data from the publishers (sink nodes). However, in data dissemination, it is the sink nodes initiating the dissemination process to push data to the normal nodes in the network. To leverage the methods in publish/subscribe systems, appropriate adjustments are needed.

3.3 Hybrid Schemes

In the existing literature, different categories have definite boundaries and no trade-off has been analyzed before. Recently, a hybrid scheme [44, 45], combining non-negotiation and negotiation-based schemes, is proposed. Hybrid schemes are proposed to balance the trade-off between efficiency and reliability in the existing framework.

Although the negotiation scheme is essential for guaranteeing a high reliability, it brings additional control overhead. The authors in [45] conduct a motivation experiment on an indoor testbed with 40 TelosB nodes to analyze the performance of non-negotiation and negotiation-based schemes. Probabilistic flooding is used as the representative of non-negotiation schemes, and Deluge (see Sect. 3.2) is the representative of negotiation-based schemes. The network topology is grid, and the distance between two adjacent nodes on the grid is 20 cm. The transmission power is set to the minimum for obtaining the multi-hop dissemination. The rebroadcasting probability in probabilistic flooding is set 0.9. The disseminating data size is 5 KB, packaging into $48 * 5$ packets. One flooding round is defined as the process that the sink node finishes broadcasting all the packets once. Reliability process is observed during the experiments. The network reliability is defined as the average of the ratio that a node number of unique received packets to the total number of necessary packets.

We cite the experiment results, Fig. 7 from [45], to analyze the trade-off between efficiency and reliability. Figure 7 shows the time-reliability curve for the two classes of schemes. From the results, we can find following observations: (1) for the small number of flooding rounds n, probabilistic flooding cannot achieve a high reliability. Hence, negotiation is a necessary for 100% reliability. (2) However, for a certain level low reliability, e.g., <30%, probabilistic flooding (with $n = 1, 3$) has a quite much shorter completion time, compared to Deluge. (3) For a high reliability, e.g., >80%, flooding (with $n = 15$) has a larger completion time than Deluge.

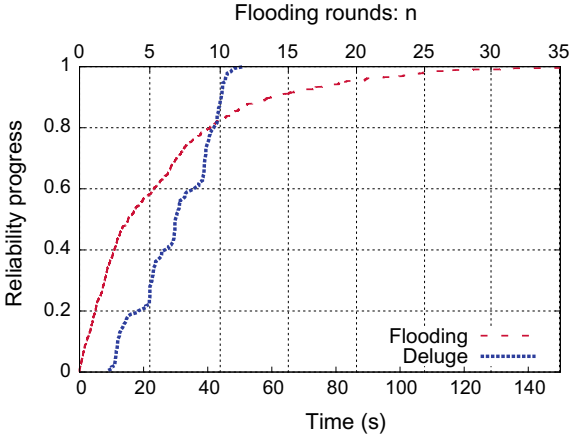


Fig. 7 Reliability progress of flooding and Deluge [45]

We summarize and analyze the observations as follows. (1) For a certain reliability, probabilistic flooding often has a much shorter dissemination time because no control message is involved to defer the data transmissions. (2) But for a higher reliability, probabilistic flooding has to increase the flooding times n to achieve, bringing inefficiency because the blind flooding without feedbacks tends to cause a large amount of redundancy. In contrast, negotiation can clearly specify the missing packets by explicit requests. The use of negotiation will effectively reduce the redundancy.

There is clear trade-off between negotiation-based schemes and non-negotiation schemes. How to balance the trade-off to get improved performance is the key point. In [44, 45], the authors propose *SurF*, a selective negotiation method, to control the transition time between two schemes to fully explore the benefit of flooding and then change to negotiation-based methods for reliability. For example, one simple hybrid scheme is flooding n times and then disseminating the remaining data by negotiation-based schemes. The key challenge is to determine when and how nodes transit from one scheme to another one. A bad transition point may result in ruin of the efforts and even bring harm to the dissemination process. For example, if the node turns to negotiation-based schemes after too many rounds of flooding, then the completion time can be longer than the completion time of using negotiation-based schemes only. In contrast, a smaller flooding times n may result in the insufficient utilization of fast dissemination. The authors model the time-reliability relationships of these two schemes [44, 45]. Based on the time-reliability model, the reliability progresses of these two schemes in the unit time could be modeled and calculated. Then the scheme that can achieve more reliability progress in the next unit time will be adopted.

4 Structure-Based Data Dissemination Schemes

Different from the structure-less schemes, structure-based schemes take advantages of the network topology and node location information for data dissemination. Even some structure-less schemes use local neighbor information, we do not regard them as structure-based methods because the neighbor information is not enforced to make the methods work. Without the prior topology information, the structure-less schemes can also work. The neighbor information is also very lightweight to learn the local information through the disseminating process. On the other hand, structure-based schemes must have a prior knowledge of the topology information, which is usually the global topology.

We further divide structure-based schemes into two sub-categories: plain and hierarchy-structure-based schemes. In plain-structure schemes, status of all nodes are equivalent in disseminating process, forming a plain structure. In plain-structure schemes, topology information is usually used to help speed the dissemination up or improve the energy efficiency. While in the hierarchy-structure schemes, nodes have different roles in the dissemination process. Normally, nodes will be divided into

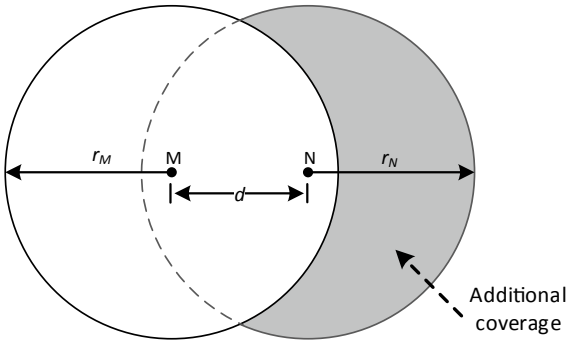


Fig. 8 Additional coverage [27]

clusters with a cluster head for each. Then the selected cluster heads form a backbone network. The data will be disseminated preferentially among the backbone network. And then the cluster heads can disseminate data in their own clusters concurrently to shorten the disseminate completion time.

4.1 Plain-Structure Schemes

Two structure-based schemes are proposed in [27] to conquer the broadcast storm problem, with the help of the location information. The first one is called *distance-based scheme*. In distance-based scheme, rebroadcasting a packet or not depends on the distance between neighbors. The benefit of an rebroadcast is defined as the additional coverage the rebroadcast can bring. The additional coverage of a broadcast is defined as the size of new covered area due to this broadcast [27]. Note that even though the mobile sensors can also increase coverage [46] or help the data collection [47], we consider the static network in the following discussion. Figure 8 is an example to illustrate additional coverage. The additional coverage of node N is shown as the gray area. Suppose the distance between M and N as d where M is a node that just broadcast the same data packet. Then if d is too small, the additional coverage of N can provide will be little. The additional coverage will become larger when d is getting larger. The relationship between d and the additional coverage area S_{add} of node N is:

$$S_{add} = \pi r^2 - 4\int_{d/2}^{r} \sqrt{r^2 - x^2}dx \geq S_T \tag{2}$$

where r_M and r_N are the communication radius of node M and N. Here, we simply consider heterogenous network that node M and N have the same radio component. Then the communication radius of node M and N is same, $r_M = r_N = r$. From the equation, we can find that if d is larger than some threshold D, then the additional coverage will be larger than some threshold. Hence, we can preset a threshold of

additional coverage which is beneficial enough for a broadcast. Then the threshold of d can be calculated and used for the judgment whether a rebroadcast should be done or not.

The second scheme is *location-based scheme*. This scheme also follows the principle that covering more additional coverage. Note that distance-based scheme has only distance information and makes the judgment of broadcast by distance threshold. When a node has more than one neighbor, distance-based scheme has to use the minimum distance to calculate S_{add} by Eq. 2 because there is no direction information available. But location-based scheme has the exact locations of nodes provided by powerful hardwares such as global positioning system (GPS). Therefore, nodes with location-based scheme can accurately calculate the S_{add} instead of estimating by distance d. In *location-based scheme*, the distance and direction information is available to calculate S_{add}, taking all the neighbors into account. Then if the calculated S_{add} is larger than the preset threshold, the node will broadcast. Otherwise, it drops the packet.

Infuse [48] is a reliable data dissemination protocol tailored to time-division multiple access (TDMA) MAC layer designs. Infuse needs localization to obtain the neighbor information of the successors in the east/south direction. In TDMA, the time period is divided into slots. Each node in a local area takes an exclusive slot. The nodes are divided into predecessors and successors based on the location information. The dissemination of Infuse is started by the start-download message from the sink. When a node receives the start-download message, it initializes the flash and notifies current running program to prepare for the upcoming dissemination process. Once receiving new data packets, a node will forward the data packet in its own time slots. During a node's transmitting slots, its successors that turn on the radios will receive the forwarded packets. Though TDMA is collision-free, packets get lost because channel fading and external interference causes the unreliable wireless links. To overcome the transmission errors, *Infuse* also designs an error recovery algorithm. A go-back-n based recovery algorithm is used in Infuse. In the go-back-n based recovery algorithm, suppose we have a window of size n. Then a node will not send packet d_i before the ACK of packet d_{i-n} is hard to guarantee the reliability.

Freshet [49] mainly focuses on energy efficiency during data dissemination. The basic design is similar to Deluge and MNP (see Sect. 3.2). What's different is that Freshet leverage network topology information to optimize the energy efficiency. In Freshet, when data transmission is not likely to happen in the neighborhood, a node will aggressively turn off the radio to save energy. The first step of data dissemination of Freshet is flooding the topology information to the whole network from sink. Then each node in the network can have an estimation about the time that data may be transmitted in its neighborhood based on this topology information. Before the estimated time of data arrival, a node just turns off its radio and go to sleep state. Such a strategy can help save energy by reducing the radio-on time when the dissemination is far away. Besides, if multiple data sources are available, it allows nodes to receive pages out of the order instead of hop-by-hop reliability guarantee, to speed up the dissemination.

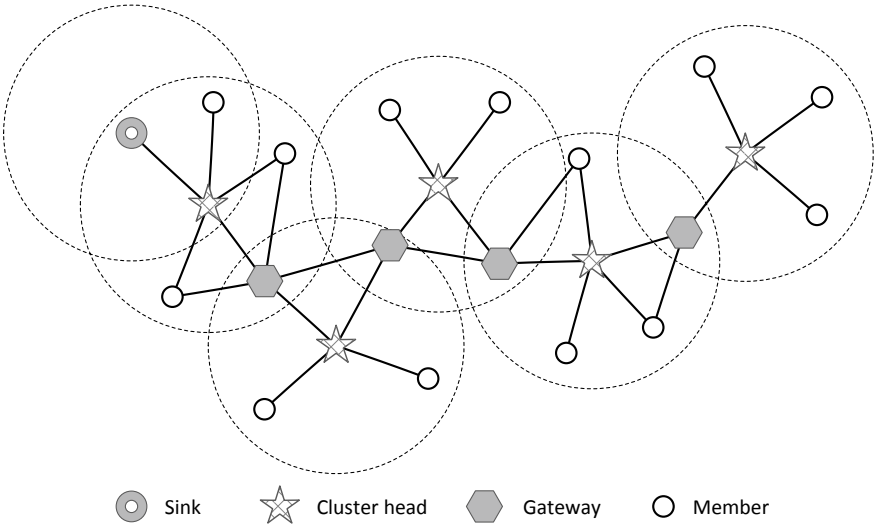


Fig. 9 An example of a clustered structure

4.2 Hierarchical-Structure Schemes

The authors in [27] also propose *cluster-based scheme* which is a hierarchical-structure scheme. Cluster-based scheme uses the topology information to divide the network into several clusters. The cluster construction algorithm is as follows. At the beginning, each node is a cluster with itself as cluster head. Then different clusters are merged together. Every node that is able to communicate with every other node in the merged cluster is a candidate of cluster head. Then the cluster head is decided by the node ID. The candidate with the minimum ID will be elected as the cluster head. The network mobility is also considered in *cluster-based scheme* [27]. When two mobile cluster heads encounter, the node with smaller ID will win the cluster head position. The other cluster will give up the head role and join into the new cluster.

Figure 9 shows an example of the cluster structure. In this cluster-based scheme, nodes are divided into three classes: cluster head as introduce above, gateway, and member. A member node keeps receiving useful data without any rebroadcast. Gateway nodes are special member nodes, locating in the overlapping region of two or more clusters. Hence, the gateways also rebroadcast packets like cluster heads do to help the dissemination process.

Sprinkler [50] is a reliable data dissemination protocol. Sprinkler uses the network topology information to construct a hierarchical structure dedicated to dissemination. Sprinkler also designs energy conserving mechanism. In Sprinkler, every node in the network is either a stationary sender or a neighbor connected to one of the stationary senders. We can find that the structure generated by above principle is equivalent

to the connected dominating set (CDS) in a graph. To save energy, less stationary senders should be used. Hence, the problem in Sprinkler is to get the structure with the minimum number of stationary senders. Such a problem is same to the problem that computing the minimum CDS problem which is NP-hard. The authors in Sprinkler propose a low complexity CDS algorithm for the sensor nodes.

After obtaining this hierarchical structure, Sprinkler performs data dissemination via two phases: streaming phase and recovery phase. In the streaming phase, only the nodes in the CDS broadcast packets. In the recovery phase, the non-CDS nodes that have missing data can send out NACK to ask for the wanted data. The CDS nodes with whole data block will respond to the requests and retransmit the requested data. TDMA is also used in Sprinkler to avoid hidden terminal problem.

Firecracker [51] is similar to Sprinkler. It combines on-demand routing and data dissemination together to accomplish rapid dissemination. Firecracker has two phases. In the first phase, sink node sends the data to the most distant nodes in the network, leveraging routing paths. Then the nodes traversed by the data from the backbone network naturally. The backbone nodes in Firecracker are the nodes in each corner or randomly selected. Once these distant nodes get the data, the second phase of Firecracker starts. In the second phase, broadcast-based dissemination is used. Data is disseminated along the paths formed in the first phase, like a string of firecrackers. The nodes in backbone networks will broadcast the data to other nodes concurrently.

CORD [52] is a reliable data dissemination protocol dedicated to the bulk data dissemination. Similar to the above three hierarchical-structure schemes, CORD also constructs a CDS dedicated to dissemination and adopts two-phase dissemination. The nodes in CORD are divided into core nodes and other normal nodes. In the first phase, data is only disseminated among the core nodes which is a connected dominating set of the network. Then the core nodes broadcast the data to other nodes concurrently. The dissemination among the core nodes can be implemented by the schemes similar to Deluge and MNP (see Sect. 3.2), which disseminates data by reliable hop-by-hop forwarding. Different from above CDS-based methods, CORD adopts a sleep scheduling algorithm in conjunction with the two-phase dissemination to save the energy.

OAP-SW [53] is a reliable dissemination protocol. The small world features are analyzed and leveraged in OAP-SW to improve the performance of data dissemination. The small world features indicate that there exist shortcuts from the sink to the other parts of the network. The idea behind OAP-SW is similar to Firecracker. First, sink quickly deliveries the data to some nodes spreading the whole network. Then all these nodes disseminate the data simultaneously to speed up the process. But in OAP-SW, nodes with more powerful hardware are added to act as the endpoints of the shortcuts, forming a heterogeneous network. Then the placement of endpoints of shortcuts is solved by designing the approximation algorithm of the minimum set cover problem. Then the first layer of the hierarchical structure is composed of the powerful nodes, and the second layer is formed by other nodes.

5 Other Techniques Used in Data Dissemination

5.1 Segmentation and Pipelining

Most data dissemination protocols (e.g., [37–40, 48, 49, 52–54]), especially the one tailored to bulk data dissemination, usually take advantage of pipelining technique to speed up the dissemination. Since wireless communication is intrinsically broadcasting, transmissions collide with each other within the communication range. But outside the communication range, the concurrent transmissions will not be overheard. Pipelining is proposed to exploit such an opportunity of spatial multiplexing.

Segmentation is proposed to cut a large data object into small pieces for more flexible dissemination. For example, in Deluge [37] (introduced in Sect. 3.2), a data object is divided into pages, each of which contains a fixed number of packets (48 packets per page in Deluge). Segmentation is usually used together with pipelining.

The segmentation is illustrated in Fig. 10. The data object is segmented into K pages. Each page has N packets. Instead of forwarding the whole data object which is only possible after completely receiving all the data, a node with segmentation technique can act as a forwarder once it receives one complete page. Then the next data page is disseminated. Segmentation helps reducing the waiting time for whole data object and prevents data staying in a single area for too long. Deluge leverages such a segmentation technique together with pipelining to speed up the dissemination process. Most of existing bulk data dissemination protocols follow the same data segmentation principle as Deluge.

Pipelining is proposed with segmentation to further exploit spatial multiplexing. To fully leverage spatial multiplexing, transmissions in different places should be performed simultaneously as much as possible. But the beneficial simultaneous transmissions should not interfere with each other. It is generally assumed that the

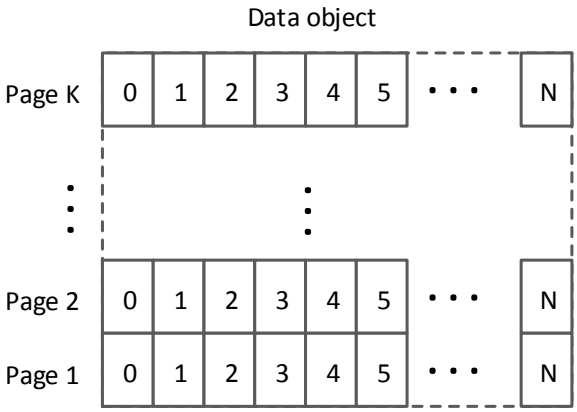


Fig. 10 Segmentation example

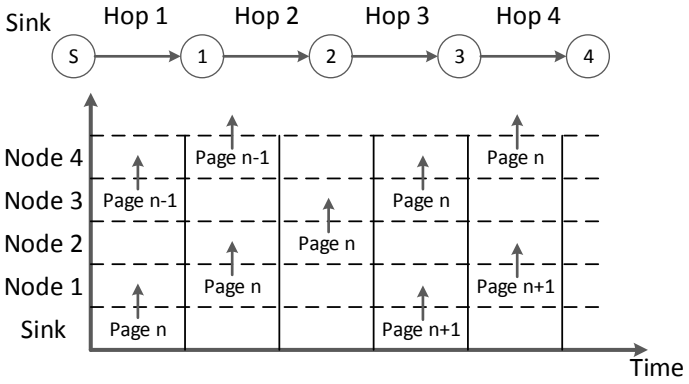


Fig. 11 Pipelining

concurrent transmissions that are three hops away will not collide with current transmission. For example, in Fig. 11, the simultaneous transmissions of sink and node 4 will not collide. They can concurrently transmit two different pages to their own neighbors, speeding up the dissemination.

5.2 Coding Technique

Coding is another promising technique used in data dissemination. Traditionally, a receiving vector on the receiver is needed to record the packet receiving status. And a to-transmit vector on the sender side is also needed to record which packets are requested by the receivers and needed to transmit in the next broadcast. This is the root cause why existing methods need NACK or ACK.

But with coding technique, a sender just broadcast the coded packets to the receivers, without the need of knowing that each receiver needs which particular packets. Then partial negotiation procedure can be cut down and the dissemination efficiency can be significantly improved. Generally speaking, the coding techniques used in data dissemination must be rateless because rateless coding technique can make different packets equivalently useful to all the receivers. To guarantee the equivalent status, all the packets are encoded into data blocks that are linearly independent. If the receivers receive enough number of linearly independent encoded packets, they can decode the original data block. When the packets are lost, there is no need for the sender to know which particular packets are lost. In methods with rateless coding, the sender can just keep sending out the encoded packets and the receivers will continue collecting encoded packets until it is enough to decode successfully.

SYNAPSE [55] is a data dissemination protocol leveraging rateless coding. The rateless coding is used in error recovery phase to improve the efficiency. SYNAPSE adopts a low computational complexity called digital fountain codes [56]. The encod-

ing and decoding are just doing the exclusive OR (XOR) operations, which is friendly to the low-power sensor nodes with limited computation resources. Redundant encoded packets will be directly transmitted for the fast error recovery. Negotiation is also used in SYNAPSE. If there is still some receivers who cannot decode the packets, the hybrid ARQ is adopted to do the negotiation. Different from the NACK/ACK in previous negotiation-based schemes that specifies the receiving vector, SYNAPSE only requires the receiver to specify the number of the additional packets it needs to perform decoding.

Rateless Deluge [57] is another work that leverages coding technique. In Rateless Deluge, two coding-based schemes are proposed: *rateless Deluge* and *ACKless Deluge*. In *rateless Deluge*, a rateless coding technique, random linear coding, is used for disseminating data. The data object will be encoded into k segments into the m linear combinations by random linear coding, where $m > k$. After encoding, the forwarder broadcasts the encoded packets. Then the receiver that gets any k messages can recovery the original data image by solving the corresponding system of linear equations.

In *ACKless Deluge*, the second scheme of [57], forward erasure correction (FEC) mechanism is further used to eliminate the need for the control packets. The random linear coding requires the receiver have more than k messages to decode the original data block. Considering unreliable wireless links, the sender must send out more than k packets. ACKless Deluge transmits the extra encoded packets in advance to reduce the probability that a receiver does not get enough packets and has to send NACK to request. But the problem is how many extra encoded packets should be transmitted. Too many encoded packets in advance will bring redundancy and waste channel resources unnecessarily. On the other hand, transmitting a small number of encoded packets in advance cannot efficiently avoid time-consuming NACK requests. Hence, ACKless Deluge integrated with FEC tries to provide enough extra encoded packets with little redundancy.

The authors in [58] propose a dissemination method for the duty-cycled WSNs. It saves energy by minimizing the number of transmissions. They propose a XORs encoding decision algorithm to minimize the number of transmissions. The effectiveness of current sleep scheduling on energy saving is analyzed and used for the decision about whether the current sleep scheduling is effective or not.

The authors in [59] provide an analytical upper bound of the completion time for coding-based methods in the WSNs with a general network topology. The results show that the completion time of coding-based dissemination methods is between $O(N)$ and $O(N^2)$ where N is the number of nodes in the network. However, in the proposed completion time model, only the network transmission latency is considered. Encoding time and decoding time are not considered.

The authors in [60, 61] propose a showcase that applying network coding to data dissemination in WSNs. They analyze the effectiveness of coding technique for small values, specifically for a single value dissemination. Traditional works usually focus on bulk data dissemination with coding technique. In [60, 61], CodeDrip is proposed to validate the performance of using network coding for small values. The results

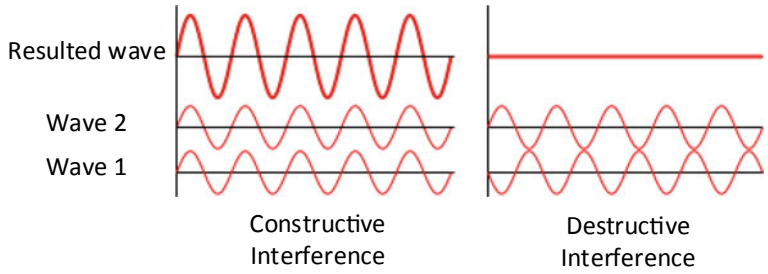


Fig. 12 Constructive interference and destructive interference

show that CodeDrip is faster, smaller, and sends fewer messages than traditional dissemination protocols for small values such as Drip [13] and DIP [14].

5.3 Constructive Interference

Recently, there is a trend of exploiting interference instead of avoiding it, such as concurrency [62–66]. Some data dissemination methods that leverage constructive interference are proposed. The proposed methods significantly change the working framework of traditional dissemination methods.

First of all, we introduce constructive interference [67]. As shown in Fig. 12, when two waves with same frequency meet at the same point, the amplitude of resulted wave is the sum of the individual amplitudes. Traditionally, wave 2 is regarded as interference when wave 1 is the interested signal. However, we can find that under constructive interference, wave 1 is amplified when wave 2 is same to wave 1. On the contrary, if wave 2 arrives at phase 180°, the crest of wave 1 meets the trough of wave 2, then destructive interference occurs and the resulted wave is interfered most seriously.

Another physical phenomenon is capture effect which can be also used in WSNs [68–71]. Capture effect is also called co-channel interference tolerance. In capture effect, the receiver is able to correctly receive a strong signal from one sender in spite of the significant interference from other senders.

Glossy [72] is a recent work that designs and implements a fast flooding scheme that leverages constructive interference to improve the performance of existing time synchronization [73]. Glossy designs a novel flooding architecture for data dissemination in WSNs. Glossy exploits CI of IEEE 802.15.4 symbols for the fast network flooding. As introduced in the previous sections, classic flooding is prone to have the broadcast storm problem. One possible approach is to schedule the broadcasts to make them interference free. However, to design such a schedule is a NP-complete problem [74].

Glossy analyzes the underlying reason of CI and intentionally makes it happen beneficially. Glossy requires strictly simultaneous transmissions of the same packet to make them interfere constructively. Then the artificial capture effects help receiver to decode the packet. Glossy disseminates data as follows. The source node initially broadcasts the first packet. Neighbors retransmit the packet right after receiving the packet to meet the conditions of CI which are (1) simultaneous transmissions; (2) same packet. Then the source continues flooding out the data packets after a certain safety time period which is enough for the previous packet floods out the communication range. By this way, Glossy can disseminate the data with a quite high reliability without any control overhead involved in the process.

However, leveraging CI faces several challenges: (1) CI solely cannot provide reliability guarantee; (2) timely error recovery needs careful design because CI keeps the radio too busy to send back ACK or NACK; (3) CI requires more than one reliable transmitters transmit the same packet at the same time. However, the reliable links in real deployment are not always available to provide efficient constructive interference.

A recent work called *Wireless Bus* [75] is proposed. In this work, the network is configured as a bus to do the data transmission without definition of unicast or broadcast. Patching it with the reliability guarantee mechanism is another possible way to do the data dissemination by leveraging CI.

Splash [76] is a dissemination protocol for large data objects in WSNs. Splash combines many techniques to achieve fast and reliable data dissemination. The constructive interference broadcast and multiple-channel pipelining are integrated to eliminate contention overhead among nodes. Without contention, the dissemination completion time can be significantly reduced. Besides, opportunistic overhearing, channel cycling, and XOR coding are also used to ensure the high reliability.

Pando [77, 78] is completely contention-free data dissemination protocol that integrates coding technique and constructive interference. With Pando, data are encoded by Fountain codes. The network is first divided into multiple parallel pipelines, based on constructive interference and channel diversity. And then the sink disseminates the rateless stream of encoded packets along the fast and parallel pipelines.

6 Performance Evaluation

The performance of data dissemination methods in WSNs is crucial to the stability and lifetime of the systems. The reason will not be repeated in details because it has been explained in Sect. 2.2. The evaluation is usually done by (1) testbed; (2) simulation. In this section, we introduce the performance metrics and then discuss the advantages and disadvantages of different schemes.

6.1 Performance Metrics

To fulfill the requirements of data dissemination, researchers propose several performance metrics to measure the satisfaction degree. We summary the common performance metrics as follows.

Reliability is the fundamental requirement as well as the essential performance metric in data dissemination. Measuring reliability usually calculates the ratio of the size of data that nodes in network need to receive, to the size of whole data that all nodes need to receive. For example, if the size of disseminating data is 10 KB and 1000 nodes exist in the network, then the size of data that all nodes need to receive is 10,000 KB. If the size of totally received data for all the 1000 nodes in network is 9,800 K according to a protocol, then the reliability of this protocol is 98%. Though sometimes the reliability is not strictly required to be 100%, the 100% reliability is a desired performance. Actually, it is usually required to be a high probability near 100% even if not 100%.

The negotiation-based schemes can achieve a 100% reliability because the negotiation process guarantees that every node gets the latest data eventually in finite time. On the other hand, non-negotiation schemes cannot achieve the 100% reliability in a limited time period because no negotiation helps the nodes with missing data to precisely and actively ask for the wanted data.

Time efficiency is crucial for system functionality. Completion time is one of the most important metrics for data dissemination to measure the time efficiency. It is crucial to the overall system performance. The time-efficiency requirement demands dissemination complete as quickly as possible. The completion time of data dissemination is defined as the time period from the first packet that the initial node sends out to the last packet that the last node in the network sends out or receives. In the non-negotiation schemes, the last node finishing broadcasting the data is the end of the dissemination. In the negotiation-based schemes, the last node receiving all the data is the end of the dissemination.

Energy efficiency is always the key concern for almost every protocol in WSNs. The less energy a protocol consumes, the longer network lifetime is. Fulfilling the requirements with minimum energy consumption is always desired. On the low-power sensor node platform, radio activities are known as the major energy consumption. Take Mica2 node as example, the energy consumption of some related operations is shown in Table 1 [28]. We can find the current draw when radio works is much larger than the other activities of nodes. Therefore, to reduce the energy consumption to the minimum, the radio-one time should be controlled as short as possible. In negotiation process, the nodes need to turn on their radios for overhearing control messages.

Memory usage is another metric to evaluate dissemination performance. Only limited memory resource is available on a sensor node. During the dissemination, memory of certain size is reserved for dissemination data and other necessary information. Generally speaking, segmentation technique can take the memory usage under control. Since the dissemination in segmentation is page by page, the required

Table 1 Energy consumption on Mica2 platform

Operations	Power consumption (*nAh*)
Read a data block from EEPROM	1.261
Write a data block to EEPROM	85.449
Send one packet	20.000
Receive one packet	8.000
Idle listen for 1 ms	1.250

memory size is the size of one-page data and other necessary information such as system parameters, neighbor information. Without the segmentation, the nodes have to store the data into flash storage if the whole data cannot be stored in memory. However, storing data in flash will cause extra delay when a node wants to send out the data because reading from the flash is slower than reading from memory. Besides, the reading and writing on flash also consume additional energy. Hence, for the bulk data dissemination, segmentation is usually adopted.

6.2 Performance Comparisons

In this section, we compare different schemes reviewed in this chapter. The detailed performance comparisons of these schemes are omitted because the experiment results reported in different papers cannot compare fairly. The results are obtained under the different experiments or simulation settings vary in different works. Therefore, we just give some general comparisons between different categories as follows.

Structure-less schemes to structure-based schemes. Structure-less schemes can be employed in a large-scale system easily because no additional information is needed. They can be very flexible to adapt to the dynamic networks with a good scalability. On the other hand, it is obvious that structure-based schemes are not easy to suit for the dynamic networks. Each change of the network environment can result in a total reconstruction of the dedicated structure which is of heavy overhead. Dedicated structure has side effects. Even though the structure-based schemes reduce control overhead during the dissemination process, they bring about another overhead of maintaining the dedicated structure.

Non-negotiation schemes to negotiation-based schemes. Non-negotiation schemes disseminate data quite quickly because no negotiation defers the transmissions of data packets. Besides, non-negotiation schemes do not have the transmission overhead of control messages brought by negotiation. The disadvantage of non-negotiation schemes is that they cannot guarantee the reliability. If high reliability is needed, the ARQ mechanism is necessary. However, the ACK/NACK messages of ARQ mechanism will incur ACK implosion problem and the broadcast storm problem. On the other hand, negotiation-based schemes sacrifice time efficiency for the high

reliability. The negotiation strategy effectively eliminates the redundant transmissions and guarantees the reliability by explicit control messages. The disadvantage of negotiation is time-costly. In recent experiment results reported in [44], the control time incurred mainly by negotiations takes around 70% of the total completion time.

Plain structures to hierarchical structures. The plain-structure schemes are more flexible, compared to the hierarchical-structure schemes. There is no cost for acquiring the topology or location information. Some of the plain-structure schemes only need the local structure information. The cost of acquiring local structure inform is limited. Another advantage of the plain structure is its convenience to construct and maintain. Compared to the plain structures, hierarchical structures can speed up the dissemination process by dividing the dissemination into two phases and broadcasting concurrently with little interference. Hierarchical-structure schemes are expected to be more efficient in the relatively stationary networks but not in the dynamic networks. If the networks are dynamic, then hierarchical-structure schemes need to reconstruct the disseminating structure once the network changes, which is too costly to maintain.

Whole image to segmentation. When data block is in a large size, disseminating the whole data hop by hop is time-consuming. By the segmenting and pipelining, the completion time will be much shorter due to the spatial multiplexing. Except some early works, segmenting and pipelining are widely adopted in the existing works.

Literature [28] and [79] also give some comparisons about the related protocols. In [80], the completion time of negotiation-based schemes is modeled and measured. The dissemination delay in the low-duty-cycle network is analyzed in [81].

7 Open Issues

Security should be considered in some application systems such as monitoring system for military use. However, the research on security in data dissemination is limited. The traditional encryption algorithms are too complex to operate on nodes. Simple alternative methods are proposed. In [82], the authors changes the file system management, reboot mechanism, and bootloader on iMote2 sensor nodes to protect the system from being reprogrammed by an unauthorized third party. The authors in [83] propose a method to achieve confidentiality in the multi-hop data dissemination. Spurious data images from the adversary are prevented by the proposed method. However, these works still require expensive resources and complicated algorithms. To keep real deployed WSNs secure during data dissemination, more effort is needed for designing secure methods more easy and more effective to use.

Heterogeneous networks should be considered in the future development of data dissemination. The boom of Internet of things (IoT) brings prosperity of smart devices. More and more IoT and WSN devices will deployed in our living environment to provide convenience to daily life [84, 85]. It is becoming more and more common many devices from different application systems are deployed in a common area. Even in the same application system, to satisfy different application require-

ments, nodes equipped with various sensors are used. Hence, heterogenous network is an unavoidable network situation.

However, data dissemination among the heterogeneous networks is not well studied. The authors in [86] study the code dissemination problem in heterogeneous WSNs. They formulate the problem as a minimum non-leaf nodes Steiner tree problem. The authors then propose a multicast protocol HSR with an approximate ratio $\ln |R|$, where R is the set of all destinations. Sprinkler [50] can also be applied to heterogeneous networks. However, these methods simply treat the heterogeneous networks as separated networks. When performing data dissemination, they first separate nodes into different sets based on the network nodes belong to. Then the data is disseminated in each separated network.

Recently, researchers propose enabling direct communication among heterogenous devices with incompatible radios [87–93]. With the ability of cross-technology communication, how to disseminate data among heterogenous devices is an quite interesting open issue. More effects are needed to improve the efficiency of data dissemination methods in the heterogeneous networks.

8 Conclusion

Data dissemination is a crucial building block for many other system services. During the long lifetime of a WSN system, it is necessary to remotely control the nodes, fix bugs, reconfigure system parameters, and upgrade the software for a better and more reliable system performance. Data dissemination over multi-hops is desired to facilitate such tasks.

Based on using a dedicated structure in the data dissemination or not, we divide existing approaches into two categories: structure-less and structure-based schemes. Structure-less scheme is further divided into negotiation-based and non-negotiation scheme according to that whether or not a negotiation mechanism is used. Structure-based schemes are further divided into plain-structure schemes and hierarchical-structure schemes. We review and compare these schemes in depth. We also elaborate the requirements and challenges of data dissemination in WSNs. We introduce some related and promising techniques such as coding and constructive interference. We also discuss the common performance metrics of data dissemination. We compare different categories and present the corresponding merits and demerits. Finally, we discuss some possible open issues based on the emerging techniques and development trends of WSNs.

Even though plentiful literature exists, there is still space to further improve the data dissemination in WSNs. Besides, some emerging techniques are very promising for changing the working framework of data dissemination. There are still some open issues that need further investigation to design an efficient protocol that can be widely adopted in real deployed systems.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Comput. Netw. **38**(4), 393–422 (2002)
2. Krishnamachari, B.: Networking Wireless Sensors. Cambridge University Press (2005)
3. Kim, S., Pakzad, S., Culler, D., Demmel, J., Fenves, G., Glaser, S., Martin, T.: Health monitoring of civil infrastructures using wireless sensor networks. In: Proceedings of ACM/IEEE IPSN, pp. 254–263, Apr 2007
4. Xu, N., Rangwala, S., Chintalapudi, K.K., Ganesan, D., Broad, A., Govindan, R., Estrin, R.: A wireless sensor network for structural monitoring. In: Proceedings of ACM SenSys, pp. 13–24, Nov 2004
5. Mo, L., He, Y., Liu, Y., Zhao, J., Tang, S., Li, X.Y., Dai, G.: Sustainable sensing in the forest. In: Proceedings of ACM SenSys, Canopy Closure Estimates with Greenorbs, pp. 99–112, Nov 2009
6. Mao, X., Miao, X., He, Y., Li, X., Liu, Y.: CitySee: Urban CO_2 monitoring with sensors. In: Proceedings of IEEE INFOCOM, pp. 1611–1619 (2012)
7. Liu, Y., Mao, X., He, Y., Liu, K., Gong, W., Wang, J.: CitySee: not only a wireless sensor network. IEEE Netw. **27**(5), 42–47 (2013)
8. Li, M., Liu, M.: Underground coal mine monitoring with wireless sensor networks. ACM Trans. Sensor Netw. (TOSN) **5**(2), 10 (2009)
9. Li, M., Liu, M., Chen, L.: Nonthreshold-based event detection for 3d environment monitoring in sensor networks. IEEE Trans. Knowl. Data Eng. **20**(12), 1699–1711 (2008)
10. Polastre, J., Szewczyk, R., Mainwaring, A., Culler, D., Anderson, J.: Analysis of wireless sensor networks for habitat monitoring. In: Wireless Sensor Networks, pp. 399–423 (2004)
11. Ma, Q., Liu, K., Miao, X., Liu, Y.: Sherlock is around: Detecting network failures with local evidence fusion. In: Proceedings of IEEE INFOCOM, pp. 792–800, Mar 2012
12. Liu, K., Ma, Q., Zhao, X., Liu, Y.: Self-diagnosis for large scale wireless sensor networks. In: Proceedings of IEEE INFOCOM, pp. 1539–1547, Apr 2011
13. Tolle, G., Culler, D.: Design of an application-cooperative management system for wireless sensor networks. In: Proceedings of EWSN, pp. 121–132 (2005)
14. Lin, K., Levis, P.: Data discovery and dissemination with dip. In: Proceedings of IEEE IPSN, pp. 433–444 (2008)
15. Dong, W., Liu, Y., Chen, C., Gu, L., Wu, X.: Elon: enabling efficient and long-term reprogramming for wireless sensor networks. ACM Trans. Embed. Comput. Syst. **13**(4), 17–43 (2014)
16. Mottola, L., Picco, G.P.: Programming wireless sensor networks: fundamental concepts and state of the art. ACM Comput. Surv. **43**(3), 19:1–19:51 (2011)
17. Werner-Allen, G., Lorincz, K., Ruiz, M., Marcillo, O., Johnson, J., Lees, J., Welsh, M.: Deploying a wireless sensor network on an active volcano. Int. Comput. **10**(2), 18–25 (2006)
18. Guo, J., He, Y., Zheng, X.: PanGu: towards a software-defined architecture for multi-function wireless sensor networks. In: Proceedings of IEEE ICPADS (2017)
19. Zeng, D., Li, P., Miyazaki, T., Guo, S., Hu, J., Xiang, Y.: Evolution of software-defined sensor networks. IEEE Trans. Comput. **64**(11), 3128–3139 (2015)
20. Cao, C., Luo, L., Gao, Y., Dong, W., Chen, C.: TinySDM: software defined measurement in wireless sensor networks. In: Proceedings of IEEE/ACM IPSN (2016)
21. Li, Z., Li, M., Liu, Y.: Towards energy-fairness in asynchronous duty-cycling sensor networks. ACM Trans. Sens. Netw. **10**(3), 38 (2014)
22. Zheng, X., Cao, Z., Wang, J., He, Y., Liu, Y.: ZiSense: towards interference resilient duty cycling in wireless sensor networks. In: Proceedings of ACM SenSys (2014)
23. Zheng, X., Cao, Z., Wang, J., He, Y., Liu, Y.: Interference resilient duty cycling for wireless sensor networks under co-existing environments. IEEE Trans. Commun. **65**(7), 2971–2984 (2017)
24. Cao, Z., He, Y., Ma, Q., Liu, Y.: L2: lazy forwarding in low duty cycle wireless sensor networks. IEEE/ACM Trans. Netw. **23**(3), 922–930 (2015)

25. Hill, J.L., Culler, D.E.: Mica a wireless platform for deeply embedded networks. IEEE Micro. **22**(6), 12–24 (2002)
26. Polastre, J., Szewczyk, R., Culler, D.: Telos: enabling ultra-low power wireless research. In: Proceedings of ACM/IEEE IPSN, pp. 364–369, Apr 2005
27. Ni, S.Y., Tseng, Y.C., Chen, Y.S., Sheu, J.P.: The broadcast storm problem in a mobile ad hoc network. In: Proceedings of ACM MobiCom, pp. 153–167, Aug 1999
28. Wang, Q., Zhu, Y., Cheng, L.: Reprogramming wireless sensor networks: challenges and approaches. IEEE Netw. **20**(3), 48–55 (2006)
29. Wang, G., Zhang, S., Wu, K., Zhang, Q., Ni, Lionel M.: TiM: fine-grained rate adaptation in WLANs. IEEE Trans. Mob. Comput. **15**(3), 748–761 (2016)
30. Tseng, Y.C., Ni, S.Y., Shih, E.Y.: Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. IEEE Trans. Comput. **52**(5), 545–557 (2003)
31. Haas, Z.J., Halpern, J.Y., Li, L.: Gossip-based ad hoc routing. In: Proceedings of IEEE INFOCOM, pp. 479–491, June 2002
32. Levis, P., Patel, N., Culler, D., Shenker, S.: Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In: Proceedings of USENIX NSDI, March 2004
33. Kyasanur, P., Choudhury, R.R., Gupta, I.: Smart gossip: an adaptive gossip-based broadcasting service for sensor networks. In: Proceedings of IEEE MASS, pp. 91–100, Oct 2006
34. Heinzelman, W.R., Kulik, J., Balakrishnan, H.: Adaptive protocols for information dissemination in wireless sensor networks. In: Proceedings of ACM MobiCom, pp. 174–185, Aug 1999
35. Kulik, J., Heinzelman, W., Balakrishnan, H.: Negotiation-based protocols for disseminating information in wireless sensor networks. Wireless Netw. **8**(2/3), 169–185 (2002)
36. Heidemann, J., Stathopoulos, T., Estrin, D.: A remote code update mechanism for wireless sensor networks. Technical report (2003)
37. Hui J.W., Culler, D.: The dynamic behavior of a data dissemination protocol for network programming at scale. In: Proceedings of ACM SenSys, pp. 81–94, Nov 2004
38. Kulkarni, S.S., Wang, L.: MNP: Multihop network reprogramming service for sensor networks. In: Proceedings of IEEE ICDCS, pp. 7–16, June 2005
39. Dong, W., Liu, Y., Wang, C., Liu, X., Chen, C., Bu, J.: Link quality aware code dissemination in wireless sensor networks. In: Proceedings of IEEE ICNP, pp. 89–98, Oct 2011
40. De, P., Liu, Y., Das, S.K.: Remo: an energy efficient reprogramming protocol for mobile sensor networks. In: Proceedings of IEEE PerCom, pp. 60–69, March 2008
41. Sarkar, R., Zhu, X., Gao, J.: Double rulings for information brokerage in sensor networks. IEEE/ACM Trans. Netw. **17**(6), 1902–1915 (2009)
42. Kündig, S., Leone, P., Rolim, J.: A distributed algorithm using path dissemination for publish-subscribe communication patterns. In: Proceedings of the 14th ACM International Symposium on Mobility Management and Wireless Access, pp. 35–42 (2016)
43. Sheltami, T.R., Al-Roubaiey, A.A., Mahmoud, A.S.H.: A survey on developing publish/subscribe middleware over wireless sensor/actuator networks. Wireless Netw. **22**(6), 2049–2070 (2016)
44. Zheng, X., Wang, J., Dong, W., He, Y., Liu, Y.: Survival of the fittest: data dissemination with selective negotiation in wireless sensor networks. In: Proceedings of IEEE MASS, pp. 443–451, Oct 2013
45. Zheng, X., Wang, J., Dong, W., He, Y., Liu, Y.: Bulk data dissemination in wireless sensor networks: analysis, implications and improvement. IEEE Trans. Comput. **65**(5), 1428–1439 (2016)
46. Li, M., Cheng, W., Liu, K., He, Y., Li, X., Liao, X.: Sweep coverage with mobile sensors. IEEE Trans. Mob. Comput. **10**(11), 1534–1545 (2011)
47. Li, Z., Liu, Y., Li, M., Wang, J., Cao, Z.: Exploiting ubiquitous data collection for mobile users in wireless sensor networks. IEEE Trans. Parallel Distrib. Syst. **24**(2), 312–326 (2013)
48. Arumugam, M.U.: Infuse: a TDMA based reprogramming service for sensor networks. In: Proceedings of ACM SenSys, pp. 281–282, Nov 2004

49. Krasniewski, M.D., Panta, R.K., Bagchi, S., Yang, C.L., Chappell, W.J.: Energy-efficient on-demand reprogramming of large-scale sensor networks. ACM Trans. Sens. Netw. **4**(1), 2 (2008)
50. Naik, V., Arora, A., Sinha, P., Zhang, H.: Sprinkler: a reliable and energy efficient data dissemination service for wireless embedded devices. In: Proceedings of IEEE RTSS, Dec 2005
51. Levis, P., Culler, D.: The firecracker protocol. In: Proceedings of ACM SIGOPS European workshop, p. 3, Sept 2004
52. Huang, L., Setia, S.: Cord: energy-efficient reliable bulk data dissemination in sensor networks. In: Proceedings of IEEE INFOCOM, pp. 574–582, Apr 2008
53. Maia, G., Guidoni, D.L., Aquino, A.L.L., Loureiro, A.A.F.: Improving an over-the-air programming protocol for wireless sensor networks based on small world concepts. In: Proceedings of ACM MSWiM, pp. 261–267, Oct 2009
54. Panta, R.K., Khalil, I., Bagchi, S.: Stream: low overhead wireless reprogramming for sensor networks. In: Proceedings of IEEE INFOCOM, pp. 928–936, May 2007
55. Rossi, M., Bui, N., Zanca, G., Stabellini, L., Crepaldi, R., Zorzi, M.: Synapse++: code dissemination in wireless sensor networks using fountain codes. IEEE Trans. Mob. Comput. **9**(12), 1749–1765 (2010)
56. MacKay, D.J.C.: Fountain codes. IET Commun. **152**(6), 1062–1068 (2005)
57. Hagedorn, A., Starobinski, D., Trachtenberg, A.: Rateless deluge: over-the-air programming of wireless sensor networks using random linear codes. In: Proceedings of ACM/IEEE IPSN, pp. 457–466, Apr 2008
58. Wang, X., Wang, J., Xu, Y.: Data dissemination in wireless sensor networks with network coding. EURASIP J. Wireless Commun. Netw. **2010**(1), 465–915 (2010)
59. Firooz, M.H., Roy, S.: Data dissemination in wireless networks with network coding. IEEE Commun. Lett. **17**(5), 944–947 (2013)
60. dos Santos Ribeiro, N., Jr., Tavares, R.C., Vieira, M.A.M., Vieira, L.F.M., Gnawali, O.: CodeDrip: improving data dissemination for wireless sensor networks with network coding. Ad Hoc Netw. **54**, 42–52 (2017)
61. dos Santos Ribeiro, N., Jr., Tavares, R.C., Vieira, M.A.M., Vieira, L.F.M., Gnawali, O.: Codedrip: data dissemination protocol with network coding for wireless sensor networks. In: Proceedings of EWSN, pp. 34–49 (2014)
62. Cao, Z., Wang, J., Liu, D., Zheng, X.: Chase: taming concurrent broadcast for flooding in asynchronous duty cycle networks. In: Proceedings of IEEE ICNP (2016)
63. Jin, M., He, Y., Zheng, X., Fang, D., Xu, D., Xing, T., Chen, X.: Smoggy-link: fingerprinting interference for predictable wireless concurrency. In: Proceedings of IEEE ICNP (2016)
64. Liu, D., Hou, M., Cao, Z., He, Y., Ji, X., Zheng, X.: COF: exploiting concurrency for low power opportunistic forwarding. In: Proceedings of IEEE ICNP (2015)
65. Cao, Z., Liu, D., Wang, J., Zheng, X.: Chase: taming concurrent broadcast for flooding in asynchronous duty cycle networks. IEEE/ACM Trans. Netw. **25**(5), 2872–2885 (2017)
66. Ji, X., He, Y., Wang, J., Wu, K., Liu, D., Yi, K., Liu, Y.: On improving wireless channel utilization: a collision tolerance-based approach. IEEE Trans. Mob. Comput. **16**(3), 787–800 (2017)
67. Michael, K., Roger, W.: The capture effect in fm receivers. In: Proceedings of DCOSS, pp. 206–215 (2016)
68. Leentvaar, K., Flint, J.: The capture effect in fm receivers. IEEE Trans. Commun. **24**(5), 531–539 (1976)
69. Whitehouse, K., Woo, A., Jiang, F., Polastre, J., Culler, D.: Exploiting the capture effect for collision detection and recovery. In: IEEE EmNetS-II, pp. 45–52, May 2005
70. Wang, Y., He, Y., Mao, X., Liu, Y., Li, X.: Exploiting constructive interference for scalable flooding in wireless networks. IEEE/ACM Trans. Netw. **21**(6), 1880–1889 (2013)
71. Wang, Y., Liu, Y., He, Y., Li, X., Cheng, D.: Disco: improving packet delivery via deliberate synchronized constructive interference. Trans. Parallel Distrib. Syst. **26**(3), 713–723 (2015)
72. Ferrari, F., Zimmerling, M., Thiele, L., Saukh, O.: Efficient network flooding and time synchronization with glossy. In: Proceedings of IEEE IPSN, pp. 73–84, Apr 2011

73. Maróti, M., Kusy, B., Simon, G., Lédeczi, Á.: The flooding time synchronization protocol. In: Proceedings of ACM SenSys, pp. 39–49, Nov 2004
74. Ephremides, A., Truong, T.V.: Scheduling broadcasts in multihop radio networks. IEEE Trans. Commun. **38**(4), 456–460 (1990)
75. Ferrari, F., Zimmerling, M., Mottola, L., Thiele, L.: Low-power wireless bus. In: Proceedings of ACM SenSys, pp. 1–14, Nov 2012
76. Doddavenkatappa, M., Chan, M.C. Leong, B.: Splash: fast data dissemination with constructive interference in wireless sensor networks. In: Proceedings of USENIX NSDI, pp. 269–282, Apr 2013
77. Du, W., Liando, J.C., Zhang, H., Li, M.: When pipelines meet fountain: fast data dissemination in wireless sensor networks. In: Proceedings of ACM SenSys, pp. 365–378, Nov 2015
78. Du, W., Liando, J.C., Zhang, H., Li, M.: Pando: fountain-enabled fast data dissemination with constructive interference. IEEE/ACM Trans. Netw. **PP**(99), pp. 1–14 (2016)
79. Sun, J.Z.: Dissemination protocols for reprogramming wireless sensor networks: a literature survey. In: Proceedings of IEEE SENSORCOMM, pp. 151–156, July 2010
80. Dong, W., Chen, C., Liu, X., Teng, G., Bu, J., Liu, Y.: Bulk data dissemination in wireless sensor networks: modeling and analysis. Comput. Netw. **56**(11), 2664–2676 (2012)
81. Li, Z., Li, M., Liu, J., Tang, S.: Understanding the flooding in low-duty-cycle wireless sensor networks. In: Proceedings of ICPP (2011)
82. Parthasarathy, R., Shirazi, B.A., Peterson, N., Song, W.Z., Hurson, A.: Management and security of remote sensor networks in hazardous environments using over the air programming. Inf. Syst. e-Bus. Manag. **10**(4), 521–548 (2012)
83. Tan, H., Ostry, D., Zic, J., Jha, S.: A confidential and dos-resistant multi-hop code dissemination protocol for wireless sensor networks. Comput. Security **32**, 36–55 (2013)
84. Wang, G., Zou, Y., Zhou, Z., Wu, K., Ni, L.M.: We can hear you with WiFi!. IEEE Trans. Mob. Comput. **15**(11), 2907–2920 (2016)
85. Zou, Y., Wang, G., Wu, K., Ni, L.M.: SmartScanner: know more in walls with your smartphone!. IEEE Trans. Mob. Comput. **15**(11), 2865–2877 (2016)
86. Peng, S., Li, S.S., Liao, X.K., Peng, X.Y., Xiao, N.: A scalable code dissemination protocol in heterogeneous wireless sensor networks. Sci. China Inf. Sci. **55**(6), 1323–1336 (2012)
87. Guo, X., Zheng, X., He, Y.: WiZig: cross-technology energy communication over a noisy channel. In: Proceedings of IEEE INFOCOM (2016)
88. Yin, S., Li. Q., Gnawali, O.: Interconnecting WIFI devices with IEEE 802.15. 4 devices without using a gateway. In: Proceedings of DCOSS (2015)
89. Chebrolu, K., Dhekne, A.: Esense: communication through energy sensing. In: Proceedings of ACM MobiCom (2009)
90. Zheng, X., He, Y., Guo, X.: StripComm: interference-resilient cross-technology communication in coexisting environments. In: Proceedings of IEEE INFOCOM (2018)
91. Guo, X., He, Y., Zheng, X., Yu, L., Gnawali, O.: ZigFi: harnessing channel state information for cross-technology communication. In: Proceedings of IEEE INFOCOM (2018)
92. Kim, S.M., He, T.: Freebee: cross-technology communication via free side-channel. In: Proceedings of ACM MobiCom (2015)
93. Liu, Z., He, T.: WEBee: physical-layer cross-technology communication via emulation. In: Proceedings of ACM MobiCom (2017)

A Data Fusion Algorithm for Multiple Applications in Wireless Sensor Networks

Gabriel Aquino, Luci Pirmez, Claudio M. de Farias, Flávia C. Delicato and Paulo F. Pires

Abstract Wireless sensor networks (WSN) are core components of the Internet of Things paradigm. Traditionally, WSNs are designed for a specific application. However, in the Internet of Things era, the sensing and network infrastructure should be shared by a set of applications from multiple owners. In such a scenario, the massive amount of data produced by the widely spread sensors is processed and analyzed to produce value-added information for the end user. By sharing the same infrastructure with multiple users, the application requirements (for instance, data intervals or the potential events of interest) may not be known a priori. In this chapter, we present a solution to properly integrate data from multiple applications without the knowledge about specific application requirements and uncover useful information from such data. We present Hephaestus, an entropic information fusion algorithm, which uses mean, kurtosis and skewness to apply a heuristic that divides the dataset into multiple features, for multiple applications. In our results, Hephaestus achieved high accuracy while incurring in low overhead for the resource-constrained devices of WSNs.

1 Introduction

WSNs [1, 32, 34] are the core sensing and communication infrastructures to enable the realization of the IoT paradigm [3, 47]. WSNs were initially designed using an application-specific approach, with little or no possibility of resource reuse or sharing among multiple applications. Besides lacking from strategies for reusing data and/or resources for different applications at different times, traditional WSN designs did not favor the sharing of the network for concurrent applications either. Two applications with the same requirements (such as data sampling interval, events, and states of interest) would demand deploying two pieces of code and duplicating the nodes' efforts of sensing, processing, and communication tasks.

G. Aquino · L. Pirmez · C. M. de Farias · F. C. Delicato (✉) · P. F. Pires
University at Albany, Albany, NY, USA
e-mail: fdelicato@dcc.ufrj.br

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_14

Dedicated WSNs (*fit-for-purpose*) are not the best, most cost-effective, or most practical deployment technique under a wide variety of conditions [25], e.g., for deployment of large-scale networks having thousands of nodes or covering large geographical areas or difficult terrains or even crowded urban areas. In particular, in the Internet-of-Things era, such design approach is not desirable, since the goal is to promote the sharing of the resources made available by the interconnected things among multiple applications, often in an opportunistic way. In a shared WSN infrastructure [16] instead of considering a fit-for-purpose design with the primary aim of supporting a single application that belongs to a single authority (usually the owner of the infrastructure), the communication and sensing infrastructure is shared by multiple applications that may belong to different users, thus optimizing the use of resources. Shared WSNs are a promising design solution for integrating such networks in IoT systems [48]. However, despite its potential, the adoption of shared sensor networks poses new challenges related to the management of the constrained resources of the **smart sensors**, which must be surpassed to fully take advantage of their envisioned benefits. One of such challenges regards managing the huge amount of generated data without jeopardizing the network performance while meeting the requirements of multiple applications.

WSNs generate, possibly in a continuous way, a large amount of data [49], which, when properly processed, will produce relevant information for human beings and the multiple applications sharing the infrastructure. For most applications, the major objective is to obtain the value-added knowledge produced by a direct result of the data that can be captured from the WSN nodes. Such high-level, value-added knowledge can drive new business and operations [31]. Therefore, core enablers to exploit the high potential benefits of WSNs are the mechanisms to collect, store, and mainly process the high volume of data generated by the sensors. Such mechanisms must deal with several issues, mainly raised from the nature of the data produced by sensors. The main features of such data are the high volume, heterogeneity (variety), and, in some cases, the rate at which they are produced (and thus may be processed to generate useful information).

In this chapter, we describe our proposal to leverage the massive amount of data produced by the widely spread sensors in a shared WSN infrastructure to produce value-added information for the end users. In traditional WSNs, applications and their requirements are known a priori, and the sensors are programmed to meet their specific tasks and collect their specific types of data, using the application required for data rate sampling, and whose values were generally within the application's well-defined data ranges [43]. On the contrary, in the emergent scenario we envision here, the application requirements may not be known a priori, neither the data intervals nor the potential events of interest. WSNs will be a shared infrastructure designed and deployed in a target area before any application is submitted. Repositories of sensing data will be generated but not necessarily tied to the specific semantics or needs of a target application. In this context, techniques to promote knowledge discovery from the produced sensing data are required to fully exploit the potential usage of WSNs. Such techniques are useful to reveal trends in the sampled data, uncover new patterns of monitored variables, and make predictions, thus improving

decision-making process, reducing response times, and enabling more intelligent and immediate situation awareness [7].

There are several open issues regarding translating the huge amount of sensor-generated data streams from their raw form into higher abstraction representations and making them accessible and understandable for human beings or interpretable by machines and decision-making systems. The vast research on information fusion techniques can be of great help to generate useful knowledge from such massive amount of data generated by a world of interconnected smart sensors [7]. The main goal of information fusion techniques is to enable data abstraction [28]. Using information fusion techniques, any type of data from structured to unstructured but, most importantly, time-series data can be integrated, transformed, and organized according to the user's needs.

In this chapter, we describe our proposed algorithm, called Hephaestus [2]. Hephaestus is an information fusion distributed algorithm [28], Bedworth and O'brien 2000, [44] for multiple applications in WSNs. Compared to the state of the art, its main distinct feature is its capability to integrate data from the monitored environment and infer environment features (e.g., an overheating in a certain area), to uncover value-added information about the monitored region. Hephaestus is able to summarize the data by inferring environmental features (i.e., information that characterizes a single or many phenomena in the monitored area), along with the uncertainty associated with the inference without being aware of the requirements (i.e., the data ranges, rates, and states) of any specific deployed application. To summarize the data without knowing the requirements of any application, Hephaestus uses an entropy [38] procedure for data analysis.

In our proposal, a phenomenon denotes any situation in the monitored area that generates data in a specific range that may or may not have correlation to an event of interest for an application in the real world. As more different phenomena generate data in the monitored area, the more complex is the data entropy, which reflects on the inference uncertainty [38]. The more the data entropy increases, the more difficult it is to separate and infer about the different phenomena individually [38], in other words, the more difficult to identify the different events of interest for applications running in WSNs.

As Hephaestus infers features about the different phenomena in the monitored area by analyzing the data entropy, each inferred feature (i.e., a recognized phenomenon) has an uncertainty quantifier [50]. Uncertainty is the measure of how much the inferred phenomena representation value may differ from the true value. The uncertainty of the inferred phenomena affects the quality of the extracted information. As sensor nodes are generally resource-constrained devices, Hephaestus analyzes the data entropy by a heuristics that respects this resource-constrained nature. As a key feature of Hephaestus, it uses the following statistics of the environment sampled data: (i) *mean*, (ii) *kurtosis,* and (iii) *skewness* [11, 22, 30]. These statistics are used to characterize the data because of their simplicity to be implemented in a sensor node. This contributes to save computation and energy from sensors. The computed statistical values will later be used by another fusion method or analyzed by an expert in some application domain in order to aid in a decision-making process.

The rest of this chapter is organized as follows. Section 2 describes some basic concepts necessary for understanding the proposal. Section 3 discusses related work. Section 4 presents the proposed information fusion algorithm, Hephaestus. Section 5 describes the current version of Hephaestus implementation, including details of the sensor platform used in the prototype. Section 6 presents a case study performed with Hephaestus, and Chapter "Sensor Assignment to Missions: A Natural Language Knowledge-Based Approach" presents the experiments for assessing Hephaestus in the context of the case study. Chapter "Resource Allocation and Task Scheduling in the Cloud of Sensors" outlines some final remarks.

2 Basic Concepts

In this section, we describe the basic concepts used in this chapter, which are needed for the full understanding of our proposal. Section 2.1 describes the basic concepts about information fusion, and Sect. 2.2 describes the statistical concepts used in this work.

2.1 Information Fusion for Wireless Sensor Networks

Data fusion [9] can be seen as "a multilevel process for dealing with the detection, association, correlation, and estimation of data from multiple sensors" [46]. In the WSN domain, simple data aggregation techniques (arithmetic averages, the search for maximum and minimum, among others) have been used to reduce the data traffic, thus reducing the energy consumption of sensor nodes. Data aggregation can be defined as the data combination from different source nodes using trivial functions (i.e., maximum, minimum, average) that suppress redundant messages and consequently reduce the amount of data. The efficiency of data aggregation algorithms depends on the correlation between the data generated by the different sources of information [17]. The correlation can be spatial, when the generated values by near sensors are related; temporal, when the readings of sensors change slowly over time; or semantic, when information in different data packages can be classified under the same semantic group, such as data that are generated by sensors placed in the same room. This aspect not only favors the elimination of redundancy (one of the goals of the data aggregation techniques), but also ensures data accuracy. This is important, because the data summarization may represent a loss in accuracy [28], which is a typical requirement for many WSN applications. The accuracy can be defined as the degree of proximity between the observed measurement and its real expected value. With an efficient correlation with the original data, it is possible to achieve a larger reduction in the amount of data for the same accuracy of the aggregated data.

Other two important concepts regarding the efficiency of data aggregation mechanism are degree and latency. The degree of aggregation is defined as the number of aggregated packets in a single packet transmission, while the latency can be measured as the time between received packets in a sink node and the data generated in the source nodes [17]. It is important that the relationship between these two concepts is balanced so that there is efficiency in the reduction in the data amount on the one hand, and no excessive delay in the final data delivery on the other hand.

The data fusion can be categorized according to several aspects, namely: the relationship between data sources, the level of abstraction, and the purpose of the data fusion. According to the relationship between the data sources, the data fusion can be classified as complementary, redundant, and cooperative [17], which are briefly described as follows.

Complementary. When the information provided by the sources represents pieces of a bigger scenario, the fusion can be applied to obtain a more complete amount of information of the scenario. The complementary fusion searches completeness, forming new information by joining several other sources (such as sensors that check for the presence of people in four different corners of a room, and by fusion of this information, we have the full view of the room).

Redundant. If two or more independent sources provide the same piece of information, these pieces can be merged to increase the information's reliability. The redundancy fusion can be used to increase the reliability, accuracy, and credibility of the information. In WSNs, redundancy fusion can provide high-quality information and prevent sensor nodes from transmitting identical data (various temperature sensors evaluating the temperature of an industrial boiler).

Cooperative. Two sources are cooperative when the information provided by them is merged into a new piece of information (usually more complex than the original), which, from the application's point of view, better represents the reality (a temperature sensor and a smoke sensor combining information to detect a fire).

Regarding the level of abstraction [17], data fusion can be classified into four levels as described below.

Signal. This level deals with single or multi-dimensional signals coming from the sensors (usually raw data coming from sensors). Signal can be used in real-time applications or as an intermediate step between fusions.

Pixel. This level regards for instance data fusion applied to image processing, which can be used in multimedia processing tasks.

Feature (or characteristic). This level deals with characteristics (or attributes) extracted from signals (such as the temperature of a room) and images, such as shape and speed.

Symbol. In this type of fusion, the information is a symbol that represents a decision (for example, a symbol indicating the alarm trigger action in case of fire), and therefore, this type of fusion is also known as fusion of decisions.

Yet another form of classification is based on the purpose of the fusion methods, in other words, what kind of information one seeks to extract from the collected data [28]. According to this criterion, the data fusion can be performed with different purposes such as inference, estimation, classification, aggregation, and compression.

Inference methods are often applied in decision mergers. In this case, a decision is made based on the knowledge of the perceived situation. The inference refers to a transition from a proposition that is probably true, from which truthfulness is credited as a result of an earlier inference. Classic inference methods are the Bayesian inference [4] and the theory of accumulation of beliefs by Dempster-Shafer (Dempster and Shafer 1975).

Compression and aggregation methods are only used to reduce the amount of data. Aggregation is used to solve the *implosion* and *overlapping* problems. In the former, the sensing data are duplicated in the network due to some routing strategy. *Overlapping* happens when two different nodes disseminate the same data. Compression methods are not data fusion methods per se, since they only consider the data coding strategies. The Huffman code is a representative example of data compression methods.

Examples of fusion methods [23] that are suitable in the WSN field are Bayesian inference, Dempster-Shafer, moving average filter and fault-tolerant average algorithm.

To illustrate data fusion methods in this chapter, we have chosen a simple method called moving average filter [23]. The moving average filter [23] is a traditional fusion method [17] widely used in digital signal processing because it is simple and capable of reducing signal noise [28]. The filter computes the arithmetic average of a number of entry signals to produce each point of the output signal. Given a signal z = (z1, z2, …, zn), the true signal x = (x1 x2, …, xn) is estimated by:

$$x(k) = \frac{1}{M} \sum_{i=0}^{M-1} z(k-i), \forall k \geq M \tag{1}$$

where M is the fusion window size, z = {z1, z2, z3, …, zn} is the input data, and x = {x1, x2, x3, … xn} is the data estimated by the method.

The fusion window is the most important parameter for the equation since M is used for the detection of any change in signal level; the larger the M value, the clearer the signal.

2.2 Kurtosis and Skewness Concepts

A fundamental task in many statistical analyses is to characterize the location and variability of a dataset. A further characterization of the data includes skewness and kurtosis [11, 22, 30]. In the following, we briefly explain these two statistical concepts, which are relevant for the understanding of the algorithm presented in this chapter.

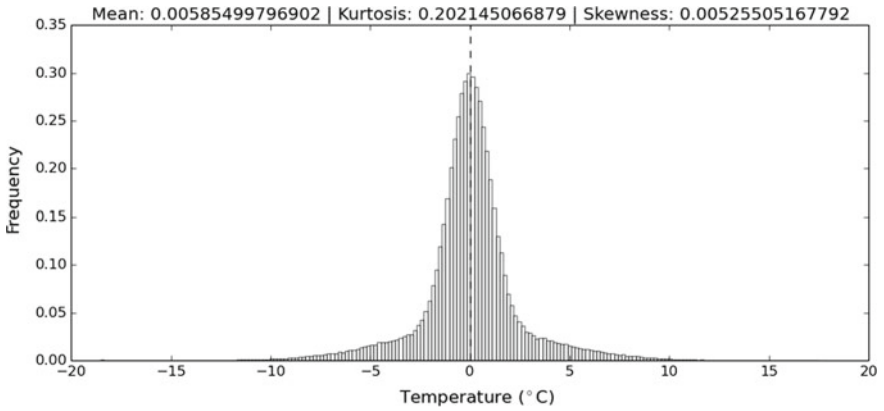


Fig. 1 A leptokurtic dataset

2.2.1 Kurtosis

In probability theory and statistics, kurtosis is a descriptor of the shape of a probability distribution. Kurtosis is a measure of whether the data are heavy-tailed or light-tailed relative to a normal distribution. That is, datasets with high kurtosis tend to have heavy tails, or outliers. Datasets with low kurtosis tend to have light tails, or lack of outliers.

The outcome of the kurtosis statistics informs how much a given data distribution is peaked or flat in relation to a normally distributed dataset [6]. For instance, by using the normal distribution as reference, we can describe if a certain dataset is *leptokurtic* (Fig. 1), *mesokurtic* (Fig. 2), or *platykurtic* (Fig. 3). A *leptokurtic* dataset has the data with higher frequency of values concentrated near the mean, which means that the center peak is much higher than the peak of a *normal distribution*. A *mesokurtic* dataset (Fig. 2) has the data concentrated around its mean like a normal distribution. A *platykurtic* dataset (Fig. 3) has a highly dispersed data, which results in a graphical dispersion of lower peak (lower kurtosis) than the curvature found in a normal distribution.

If a dataset is *leptokurtic*, the mean value properly represents the dataset. In this case, the frequency of the mean value is greater than the frequency of the mean value of a *mesokurtic* dataset. A *platykurtic* dataset indicates that there is a higher dispersion around the mean, and in this case, the frequency of the mean value is lower than the mean value of a *mesokurtic* dataset. If a dataset is *platykurtic*, the mean of the dataset may not represent well the dataset. The kurtosis statistics can be used to indicate how representative the mean is for a given dataset.

There are different ways to measure the kurtosis of a data sample. For instance, a quantile-based index can be used to calculate the kurtosis. This is the approach adopted in the proposal described in this chapter. In such an approach, for a given set of sampled data, the data are sorted from the smallest to the largest values and then the set is divided into quantiles, which are equal-sized data subsets.

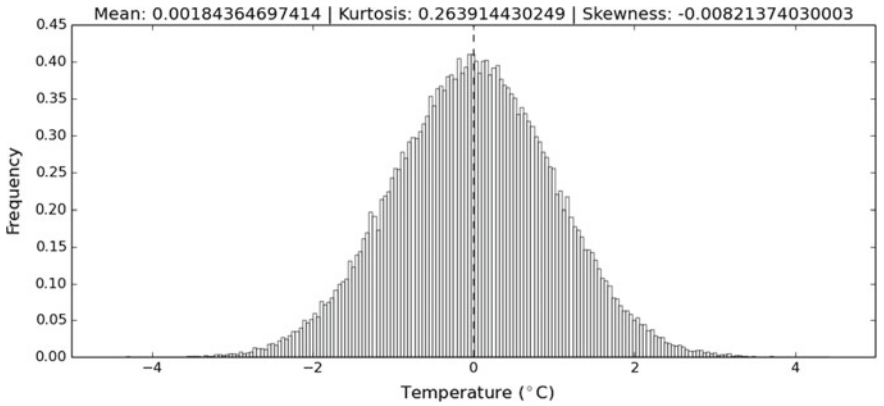


Fig. 2 A mesokurtic dataset

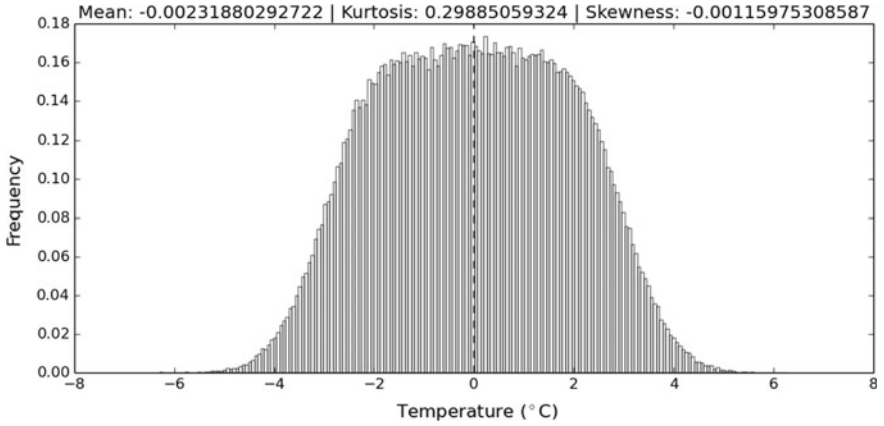


Fig. 3 A platykurtic dataset

The quantiles are the data values marking the boundaries between consecutive subsets. A quartile is a type of quantile. The first quartile (Q_1) is defined as the middle number between the smallest number and the median of the dataset. The second quartile (Q_2) is the median of the data. The third quartile (Q_3) is the middle value between the median and the highest value of the dataset. A percentile is also a type of quantile. A percentile measure is used to indicate the value below which a given percentage of observations is in a group of observations. The datum at n-percentile is the P_n value.

Using the percentile and quartile values of the environment sampled data, it is possible to measure the kurtosis of the dataset. The equation to calculate kurtosis is given in (2) [45]:

$$\mathrm{K} = \frac{1}{2}\frac{\mathrm{Q}_3 - \mathrm{Q}_1}{\mathrm{P}_{90} - \mathrm{P}_{10}} \tag{2}$$

The outcome of (2) can be [36] (i) $K = 0.263$, indicating a *mesokurtic* dataset (Fig. 2); (ii) $K < 0.263$, indicating a *leptokurtic* dataset (Fig. 1); and (iii) $K > 0.263$, indicating a *platykurtic* dataset (Fig. 3).

2.2.2 Skewness

Skewness is a measure of symmetry, or more precisely, the lack of symmetry. A distribution, or dataset, is symmetric if it looks the same to the left and right of the center point. The problem of data asymmetry was studied by Pearson [30], and it indicates that an asymmetrical frequency curve may arise in two cases: (i) in the case of homogeneous material when the tendency of deviation on one side of the mean is unequal to the tendency of deviation on the other side and (ii) when the material measured may be heterogeneous or may consist of a mixture of two or more homogeneous materials [30].

In this chapter, we refer to the Pearson's coefficient of skewness (3) that uses the median of the dataset [20]:

$$S = 3\frac{(\text{mean}(\text{dataSet}) - \text{median (dataSet)})}{\text{standardDeviation (dataSet)}} \tag{3}$$

A dataset might be (i) positively skewed (Fig. 5); (ii) negatively skewed (Fig. 4); or (iii) symmetric (Fig. 2) [6]. A (i) positively skewed dataset has a positive value of S ($S > 1$), and the tail of the dataset is longer on the right and the concentration mass of the data is set on the left of the mean (Fig. 5). A negatively skewed dataset is the opposite of a positive one, which means that the value of S is negative ($S < 1$) and the tail of the dataset is set on the left and the mass of the data is concentrated on the right of the mean (Fig. 4). A symmetric dataset has $S = 0$, and the data are concentrated toward its mean in a homogeneous way (Fig. 2). There is a moderate asymmetry when $0.15 < |S| < 1$ (Figs. 4 and 5), and there is a strong asymmetry when $|S| > 1$ (Fig. 6).

A moderate positively skewed (Fig. 5) or a moderate negatively skewed dataset (Fig. 4) indicates that there is an increase or decrease in tendency. A strong skewed dataset indicates that the dataset is formed by two or more groups of heterogeneous data (i.e., the samples were from heterogeneous materials). The skewness statistics can be used to describe patterns and tendencies of the dataset.

3 Related Work

The challenge of leveraging the massive amount of data produced by sensors to produce value-added information for the end users has promoted the publication of several recent works focused on multiple applications.

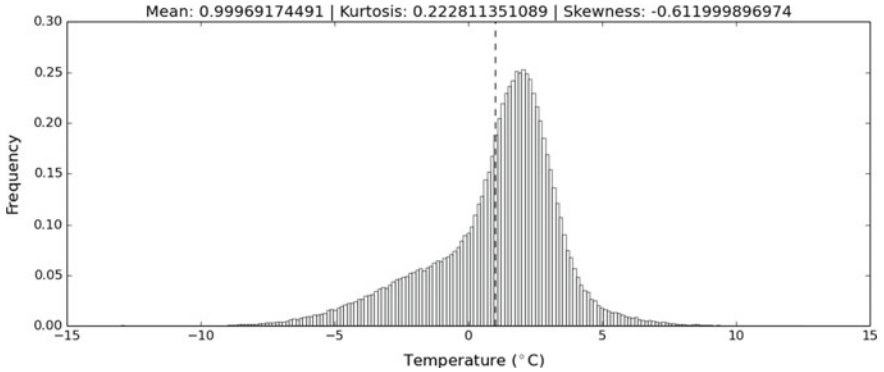


Fig. 4 Negatively skewed dataset

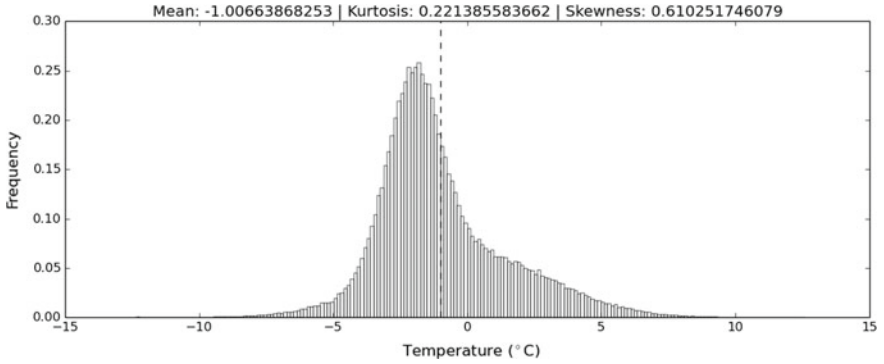


Fig. 5 Positively skewed dataset

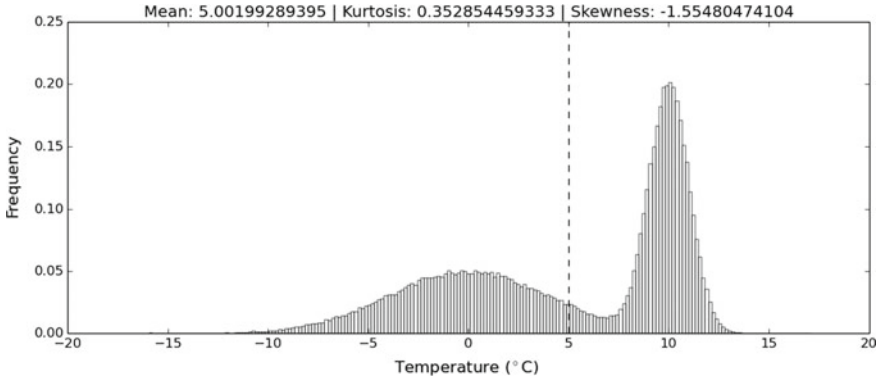


Fig. 6 Strongly skewed dataset

The works described in [18, 19, and 35] deal with the challenge of executing information fusion of data for multiple applications in WSNs, but these proposals need to be aware of the requirements of the deployed applications. First, we present the work in [19], which modifies the moving average filter (MAF) to evaluate the sensed data differently according to the importance of these data for the application. The work in [19] presents the enhanced moving average filter (EMAF), whose main idea is to weight the dataset to express the requirements (data intervals, data rates, and states) of different applications. The drawback of this approach is the necessity of knowledge of the application requirements. On scenarios where the set of applications changes quickly (as in Smart Cities, where applications may belong to anyone and be deployed anytime), EMAF becomes infeasible, as it has to be constantly reconfigured. As EMAF [19] uses raw data (measurements with low level of abstraction), higher abstraction level data (such as decisions) are not properly handled. Therefore, in a posterior work, the same authors of EMAF presented fusion methods of higher abstraction levels [18]. In [18], the authors adapt some existing fusion methods to execute information fusion of data for multiple applications in WSNs. They propose the following information fusion methods: (i) enhanced Bayesian inference (EBI), (ii) enhanced Dempster-Shafer inference (EDSI), and (iii) enhanced fault-tolerant averaging (EFTA). EBI formalizes the combination of evidences according to the rules of the Bayesian probability theory for each application [4]. EBI represents the hypothesis that an application Y will have a determined behavior given the result of an application "X", by considering the set of states of each application isolated. In EDSI, each application has its own set of hypotheses, which represents the behavior of an application. EDSI infers through the Dempster-Shafer Inference method over conditions of isolated applications and considers the belief that both applications will be in a given state simultaneous. EFTA evaluates the data intervals according to the requirements of each application, applying the traditional fault-tolerant interval method. Then, EFTA produces a combination of each evaluated data interval.

Despite dealing with information fusion for multiple applications, the proposals in [18] and [19] present a significant restriction as they need to be preconfigured with the requirements of each application deployed in the network. While EMAF [19] needs to know about the requirements of the applications to properly weight the dataset, EBI, EDSI, and EFTA [18] need to infer states for each application to present a decision that integrates the multiple applications. In scenarios where the set of applications in execution changes quickly (as in Smart Cities, where applications may belong to anyone and be deployed anytime), such works become infeasible. Hephaestus introduces an information fusion method that is not dependent on applications' requirements, since it performs the fusion procedures through an entropic data analysis.

Safia et al. [35] propose a distributed algorithm to detect phenomena, such as fires, oil spills, or poisonous gases, in a WSN environment where both the sensors are mobile and the phenomenon is dynamic (e.g., moving, growing, or shrinking). The authors assumed that the environment has no centralized server to collect and aggregate the data of the sensors. In the algorithm [35], the sensors organize themselves into disjoint groups by first electing a few sensors to be leaders of the group

and then the rest of the sensors group themselves with the nearest leader. Sensors of each group report their sensed data to their group head, which aggregates the collected data and detects local phenomena (within the area spanned by the member sensors of the group). Then, based on the order of the leader's IDs, each leader reports the detected local phenomena information to the next leader in that order, which aggregates the information and sends it to the following leader, and so on. The last leader in the order chain aggregates the information of all the detected local phenomena to discover the global phenomena. Additionally, Safia et al. [35] propose two leader election algorithms, one for electing leaders based on the global phenomena information and another for electing leaders based on the local phenomena information. Furthermore, Safia et al. [35] propose an optimization technique to reduce the energy costs of reporting the local phenomena information. In this technique, the reported local phenomena information between leaders is summarized by its convex hull representation and therefore reduces the size of the transmitted data between leaders.

The work described in [35] presents some similarities to the proposal presented in this chapter, since it is a distributed algorithm to detect phenomena in a WSN environment. But, different from our work, they follow some assumptions: (i) for [35], sensors have the same processing power, battery life, storage, and communication range in its initial phase; and (ii) sensors have prior knowledge of the "normal range of values" (non-phenomena values). Concerning the two assumptions considered in [35], the main differences from Hephaestus to such a work is that Hephaestus does not need to run in a network in which sensors have the same processing power, battery life, storage, and communication range in its initial phase and, mainly, Hephaestus does not need to be aware of any prior knowledge of the environment. These two characteristics broaden the applicability scope of our algorithm.

The works described in [5] and [27] deal with the challenge of executing information fusion of data for multiple applications in WSNs without being aware of the requirements of the deployed applications. However, the results of these information fusion algorithms are not applied in a general context; instead, they execute an information fusion procedure with the goal to produce a clustering hierarchy.

Bicocchi et al. [5] presented an algorithm to let a sensor network self-organize a virtual partitioning scheme in correspondence to spatial regions characterized by similar sensing patterns. Their proposed algorithm allows a distributed aggregation of sensorial data to take place in a per-region basis, which results in a modeling of a sensor network as a collection of virtual macro sensors, each one associated with a well-characterized region of the physical environment. Within each region, each physical sensor has the local availability of aggregated data about its region and can act as an access point to this area. The virtual macro sensor approach (ViMS) [5] includes three main aspects: (i) a self-organized region-formation algorithm, to logically split a sensor network into a set of spatial regions, each characterized by specific environmental patterns in the sensed data; (ii) localized in-network aggregation algorithms to provide each sensor in a region with aggregated information about the overall state of the region; and (iii) peculiar innovative solutions to self-adapt to transitory and dynamic situations. Such solutions ensure that both regions and aggre-

gated information within regions always reflect the current situation of the network and of the environment. In comparison with our proposal, the ViMS approach can be very effective in supporting multiple users. Another similar aspect is the capability to facilitate the data collection in dense large-scale sensor networks and enforce activities for the recognition of phenomena and situations. But, different from the ViMS approach, our work proposes an information fusion algorithm, while [5] proposes a self-organizing algorithm to partition a network in spatial regions by similar patterns. While the ViMS approach demands partitioning the network in similar patterns, our proposal does not require such a partitioning process to recognize phenomena and situations from the data, and it also provides knowledge of the locality of the several phenomena taking place over the monitored area. We consider our work as a tool that can also be used to partition the network, by using our proposal output as entry in a network partitioning algorithm.

Mus and Kr [27] proposed E-BACH: Entropy-Based Clustering Hierarchy for WSNs. They developed a clustering hierarchy to form a heterogeneous WSN with nodes operating with data collection rates determined by their entropy. Mus and Kr [27] proposed a new method that uses data quality as the primary goal of network optimization. Based on the concept of data entropy, the proposal creates a hierarchical heterogeneous network where individual nodes sample the measured quantities according to the potential information gain. The authors state that this can be used to save energy by lowering sampling rates of nodes at location with low variability of monitored variables or high correlation with other nodes. The entropy measures and their approximations outlined earlier can be used to identify clusters of sensors. Depending on the specific measures used, results of the procedure proposed by [27] would develop hierarchy of clusters according to their information contents and identify groups of sensor nodes that provide similar data. For this purpose, entropy measures and their approximations can be treated as distances (information distances) between sensor data. In short, the contribution of [27] is the proposal of a novel WSN clustering algorithm based on the data entropy of individual sensor nodes. In a similar way as our work, [27] propose a data processing algorithm that uses the data organization (entropy) to facilitate data collection without being aware of the requirements of the applications in the network. But, different from our proposal, they do not have a general-purpose information fusion process; instead, the fusion has the specific goal of clustering the network. We consider the proposal of this work—Hephaestus—as a tool that while not having been designed with the specific purpose of clustering can aid in the process of clustering a network.

Table 1 summarizes the findings of this section and divides the related works into 3 categories. The first category groups information fusion algorithms that consider multiple applications but need to be aware of the requirements of the deployed applications. **The second** category relates to information fusion algorithms that consider multiple applications, without being aware of the requirements of the deployed applications, but whose result is not applied to an application rather being used as an analysis tool of the network behavior. **The third category** relates to information fusion algorithms for multiple applications, not aware of the requirements of

Table 1 Groups of related proposals

Group	Related works
Multiple applications; requirements aware	Farias et al. [19]
	Farias et al. [18]
	Safia et al. [35]
Multiple applications; requirements agnostics; used as an analysis tool for the WSN behavior	Bicocchi et al. [5]
	Mus and Kr [27]
Multiple applications; requirements agnostics; outcome of the fusion as input for a decision	Hephaestus

the deployed applications and producing outputs that can be used as an input of a decision. Only Hephaestus is classified in this third category.

4 Hephaestus

In this section, we present and describe Hephaestus, our proposed information fusion algorithm for multiple applications in WSNs.

Hephaestus is a topology agnostic and decentralized information fusion algorithm for WSNs. Hephaestus is able to infer environment features, actually discovering information about the environment, which may also contribute to improving the energy efficiency of WSNs. Hephaestus is a decentralized algorithm since the whole information fusion process of integrating data for different applications is performed within WSNs. Therefore, there is no need of transmitting raw data to a centralized entity (base station/sink node/gateway) on a hop-by-hop basis, thus reducing the need of radio transmissions and saving energy. Hephaestus is an information fusion algorithm tailored for multiple applications since the information produced by Hephaestus is not tied to any specific application. The outcome of Hephaestus algorithm can be used by any application running in WSNs.

We consider a WSN composed of a finite set of sensors and one or more sink nodes. The network is considered heterogeneous since each sensor node has different capabilities in terms of storage, processing, sensing, and communication. Such set of sensor nodes are in charge of performing all the procedures of Hephaestus algorithm. We consider two roles for the sensor nodes, denoting their responsibilities in the network: (i) collector node (CN) and (ii) fusion node (FN). A given node may have only the CN role set, only the FN role, or even both CN and FN roles simultaneously. A CN is considered the basic "sensing, processing, and deciding unit" in WSNs. Such nodes are equipped with at least one physical sensing device. A FN is responsible for executing the information fusion algorithms. Such algorithm is composed of two procedures: (i) the local information fusion procedure (LIFH) and (ii) the complementary information fusion procedure (CIFH).

The **local information fusion procedure** (LIFH) is responsible for locally receiving data and integrating data from the monitored environment, inferring environment features (characteristics of potential events of interest), actually discovering information about the environment, and forwarding the inferred features to the sink through other fusion nodes (FNs). This procedure executes an entropic data analysis, which is responsible for statistically analyzing the samples from the environment and executing a heuristics to infer environment features.

The **complementary information fusion procedure of Hephaestus** (CIFH) is classified as a complementary information fusion according to [28]. CIFH running in a given sensor is responsible for integrating the output information from LIFH running in its neighboring sensor nodes. CIFH consolidates different LIFH information, since the nodes running in LIFH have a view of the phenomena limited to their sensing range and therefore different from the actual phenomena in the monitored environment. The objective of CIFH is to fuse the data from neighboring nodes in LIFH sensors in order to have a broader view of the phenomena taking place in the network.

All FNs execute both LIFH and CIFH. If a FN receives a message from CN (i.e., a temperature sample), it executes LIFH. Else, if a FN receives a message from another FN, it executes CIFH to compose a broader view of the phenomena in the WSN monitored area.

4.1 Local Information Fusion Procedure (LIFH)

LIFH is performed immediately right after filling a fusion window (which is the dataset with the samples to be analyzed) by receiving the data collection from the neighboring nodes in CN. LIFH encompasses a sequence of steps performed in two main phases: (i) **Entropic Indexes Calculation** (EIC) and (ii) **Phenomena Discover Analysis** (PDA). Initially, the EIC phase is performed. In this phase, the data entropy [38] is analyzed through a peak analysis in the sampled data. This peak analysis is done using skewness and kurtosis [22, 30] over the collected environmental data.

Next, the PDA phase starts. This phase is responsible for analyzing the skewness and kurtosis results and for generating data samples from the previous EIC phase in order to recognize different phenomena that are taking place over the monitored environment. The skewness result is used to indicate tendencies or heterogeneity of the data; the kurtosis is used for quantifying the uncertainty, which is the measure of how much the inferred phenomena representation value may differ from the true value, about an inferred phenomenon; and the arithmetic mean to represent the mass center of the inferred phenomena. In the following sections, we present and describe the data structures and the detailed operation of LIFH.

4.1.1 Data Structures Used in the LIFH

The data structures used in LIFH are *ESD* (Environment Sampled Data), *EIS* (Entropic Indexes Structure), *LESD* (Lower Environment Sampled Data), *HESD* (Higher Environment Sampled Data), *IPhenomena* (Inferred phenomena), and *HFPS* (Hephaestus Feature Points data Structure).

(i) *ESD* data structure stores the raw sampled data obtained from the sensing units responsible for monitoring the target environment; the (ii) *EIS* is the Entropic Indexes Structure, which stores the mean, kurtosis, and skewness of the data samples stored in *ESD*; (ii) *LESD* and *HESD* are data structures used to store a subset of the data (*LESD* stores data lower than the average and *HESD* above the average) in the *ESD* data structure; (iv) the *IPhenomena* data structure stores the individual inferred phenomenon, as recognized by *LIFH*; and finally, (v) *HFPS* stores the features inferred by the procedure. *HFPS* contains the inferred phenomena, a list of *IPhenomena*.

A phenomenon denotes any situation in the monitored area (e.g., an overheating in a certain part of the area) that generates data in a specific range that may or may not have correlation to an event of interest for an application in the real world. Concerning the *IPhenomena* data structure, each inferred phenomenon is represented as the mean of the range of the phenomenon, along with its uncertainty characterizer. The more different phenomena generate data in the monitored area, the more complex is the data entropy, which reflects on the inference uncertainty [38]. The more the data entropy increases, the more the difficult it is to separate and infer about the different phenomena individually [38], in other words, the more the difficult to identify the different events of interest for applications running in WSNs. The uncertainty characterizer is generated through kurtosis analysis, in which the greater the value of kurtosis, the more the data are dispersed around its mean value, and thus, more uncertainty is present in the environment data. Each reasoned phenomenon has an uncertainty characterizer. Each inferred phenomenon is stored into the *HFPS* data structure.

In our proposal, each data feature is an inferred phenomenon and *HFPS* summarizes all the phenomena inferred by LIFH. The *HFPS* data structure is a 2-dimensional discretized representation of the inferred phenomena along with an uncertainty quantifier [50] related to the phenomenon inference. Each phenomenon is represented by the arithmetic mean of the dataset. The uncertainty quantifier indicates a behavior of the inferred phenomena. It expresses the environment data entropy, and it quantifies how difficult it is to separate and infer about the different phenomena individually. The uncertainty of the inferred phenomena affects the extracted information quality.

4.1.2 LIFH Operation

In this subsection, we present the detailed operation of LIFH. First, the Entropic Indexes Calculation (EIC) phase is performed. The data samples received from the environment are stored in *ESD* (Environment Sampled Data). After that, arithmetic

Entropic Index Calculation (EIC):
Input: The environment sampled data (*ESD*);
Output: The Entropic Indexes Structure: *EIS*;

1. Calculate and store the mean of the samples on *ESD*;
2. Calculate and store the kurtosis of the samples on *ESD*;
3. Calculate and store the skewness of the samples on *ESD*;
4. *EIS* = [*Mean* (*Me)*, *Kurtosis* (*Ku*), *Skewness* (*Sk*)];
5. PDA(*EIS*);

Phenomena Discover Analysis (PDA):
Input: The Entropic Indexes Structure: *EIS*;
Output: Hephaestus Feature Map data Structure: *HFPS*;

6. **If** the absolute value stored on *Sk* is lesser or equal to 1:
7. **If** the value stored on *Ku* is lesser or equal to 0.263:
8. *IPhenomena* = (*Me*, "Leptokurtic Set")
9. **Else**, if the value stored on *Ku* is greater than 0.263:
10. *IPhenomena* = (*Me*, "Platykurtic Set")
11. **End If**
12. HFPS.add(*IPhenomena*)
13. **Return** HFPS;
14. **Else,** if the absolute value stored on *Sk* is greater than 1:
15. *LESD* = *ESD* values lower than the value in *Me*;
16. *HESD* = *ESD* values greater than the value in *Me*;
17. HFPS.concatenate (PDA(EIC(*LESD*)));
18. HFPS.concatenate (PDA(EIC(*HESD*)));
19. **Return** HFPS;

Fig. 7 LIFH

mean, kurtosis, and skewness of the received data are calculated (lines 1, 2, and 3 in Fig. 7) and stored in the *EIS* structure (line 4 in Fig. 7).

After the EIC phase is finalized, the main phase of LIFH, namely the Phenomena Discover Analysis (PDA) starts (line 5; Fig. 7). The PDA phase is responsible for creating the *HFPS* data structure according to the entropic indexes (mean, kurtosis, and skewness) calculated in the EIC, which are stored in *EIS*. In this phase, the discovering of different phenomena from the environment data is performed by using the skewness statistics (line 6 and line 14). The skewness statistics may indicate two different types of environment data: (i) moderately skewed (line 6; Fig. 7) and (ii) strongly skewed (line 14; Fig. 7).

If the environment data are moderately skewed $(0.15 < |Sk| < 1)$, there is a single phenomenon or multiple phenomena generating data with similar data ranges in the monitored area (line 6; Fig. 7). When there is a moderate skew, the environment data can be leptokurtic or platykurtic (line 7 or 9). The shape of the peak indicates the number of occurrences of a certain situation. The more often a certain situation occurs, the more the environment data is leptokurtic (which means that as the volume

of values from a certain data range is generated, the greater is its concentration around the mean). In this case, the leptokurtic environment data indicate that the data are more concentrated. Higher peaks show that there is a well-defined data range for a given set of phenomena (lines 7 and 8; Fig. 7). For example, a high concentration of temperature samples on a 55 °C peak may indicate the presence of fire. The inferred phenomenon is classified as "Leptokurtic Set" if the kurtosis index is lesser than or equal to 0.263. In this case, the inferred phenomena representation was the previously calculated mean stored in *EIS*, and it represents well a single or a set of phenomena. A set classified as "Leptokurtic Set" guarantees that all data are closer to the mean for the phenomena that generated it; in this sense, these data have less variability. This feature is stored in *IPhenomena*.

If the environment data are platykurtic (e.g., Fig. 3), then the outcome of data fusion will have a considerable degree of uncertainty about the set of phenomena. In this case, the kurtosis index is greater than 0.263. For addressing such case, we propose using the descriptive quantifier "Platykurtic Set" and the previously calculated mean value stored in EIS data structure to characterize the inferred phenomenon stored in *IPhenomena* (lines 9 and 10; Fig. 7). A platykurtic dataset represents the case where different phenomena but with closer data ranges are taking place in the monitored area. When the environment data are platykurtic, LIFH may not correctly identify the different phenomena that generated such environment data since it is difficult to identify an isolated peak as the ones in leptokurtic and mesokurtic datasets. A set classified as "Platykurtic Set" should be properly handled to guarantee a higher data accuracy for the phenomena that generated it as it may indicate a new phenomenon creating outliers or a sensing failure, producing a multimodal set. Then, the inferred phenomenon is inserted in *HFPS,* and this data structure is returned as a result of LIFH for the analyzed data.

If the environment data are strongly skewed (**|Sk| > 1**) (e.g., Fig. 6), there are multiple disjoint phenomena occurring in the monitored area with very distant ranges (line 14; Fig. 7). In this case, LIFH divides *ESD* into two samples. The values lower than the mean are stored in *LESD* (line 15). The values greater than the mean are stored in *HESD* (line 16). After filling *LESD* and *HESD*, LIFH re-executes EIC and PDA using *LESD* and *HESD* as input (lines 17 and 18). After re-executing the EIS and the PDA phases using *LESD* and *HESD* as input, LIFH has to insert the recognized phenomena in *HFPS* (lines 12, 13, and 19).

It is important to notice that by dividing the dataset when it is strongly skewed ($|Sk| > 1$) and re-executing the EIC and the PDA phases in each half, Hephaestus divides the dataset in an iterative way until it is divided into groups of moderately skewed datasets. Hephaestus recognizes each moderately skewed dataset as a feature (an inferred phenomenon), and represents this phenomenon as its mean, and the uncertainty as its kurtosis.

After recognizing the phenomena locally, *HFPS* should now be given as input to any existing information fusion procedure to interpret and generate a decision (the datasets grouped around the identified peaks). The fusion nodes will send their decisions to the sink node where *HFPS* will be used to make decisions. The technique used to make decisions over *HFPS* in the sink node is out of the scope of this chapter.

4.2 Complementary Information Fusion Procedure (CIFH)

CIFH is a procedure that integrates data from nodes running in LIFH. CIFH is responsible for fusing similar phenomena (in terms of sensing type) inferred by neighboring nodes running in LIFH in a wide area. Since the result of the information fusion of a single node running in LIFH presents a partial vision (constrained by the sensing coverage), CIFH performs the complementary fusion of data from neighboring nodes in order to compose a broader view of the environment. CIFH starts right after receiving the first message from a node running in LIFH. CIFH operation encompasses a sequence of steps, which are performed in two main phases: (i) the setup phase and (ii) the periodical phase.

The setup phase is executed only once and occurs when a node running in CIFH receives its first message from a FN running in LIFH. If the node running in CIFH receives any message during the setup phase, such node stores the received messages in memory to be further analyzed during the periodical phase. The setup phase is responsible for setting up some initial parameters (such as freeing some data structures, setting up the node position on x-axis and y-axis, setting a tolerance time in which CIFH will accept messages to assure time correlation between the fused information, and setting the quantity of messages it will fuse in a single information) to assure its correct execution.

The periodical phase is the main phase of CIFH. It is responsible for receiving messages containing data from LIFH, and such reception occurs periodically. The periodical phase evaluates similar phenomena into a single phenomenon to reduce data transmission and avoid data misinterpretation. Similar phenomena are the ones having similar peaks (skewness and kurtosis values). Similar peaks are two or more data samples whose difference among means is less than a given threshold, and typically, the threshold is smaller than the standard deviation.

4.2.1 Data Structures Used in CIFH

CIFH has the following data structures, detailed below: FN.id, Hephaestus Feature Points data Structure (HFPS), Wide View Phenomena data Structure (WVPS), threshold unit (Tu), differences dictionary (Dd), rounds (R), timeWindow, tolerance interval (TInt), node position on Cartesian x-axis (pX), node position on Cartesian y-axis (pY), node operating range (radius). Also, CIFH uses the function *Receive_Hephaestus_Map(timeWindow, pX, pY, radius*).

The *FN.id* data structure stores the identification of the fusion node that sent a LIFH's message. The *HFPS* data structure stores all phenomena in the received message, which was inferred by a node running in LIFH. The *WVPS* data structure stores the fusion result of CIFH. The *WVPS* comprises of different phenomena marked with the identification of the node that recognized it (*FN.*id). The *Threshold unit (Tu)* data structure stores a threshold related to the unit (temperature, humidity, acceleration, etc.) of the phenomena in *HFPS*. The *Dd* data structure stores the differ-

ences between a given phenomenon P in *HFPS* and all phenomena P' in the *WVPS*. The key entry of the *Dd* is the pair (P, P'), and the value is the difference between |P − P'|. The *R* data structure stores the number of rounds during which the complementary fusion procedure will receive messages after sending the *WVPS* to the sink node. The *timeWindow* stores the acceptable time interval tolerance to summarize different messages. The *TInt* data structure stores the acceptable time tolerance for receiving messages. The value in *TInt* is used to create the time interval stored in the *timeWindow* data structure. The *Receive_Hephaestus_Map(timeWindow pX, pY, radius)* function uses the values of *timeWindow, pX, pY,* and *radius* to discard or accept a message. The function **discards a** message when: (i) its time stamp is out of the interval in the *timeWindow* and (ii) it was sent by a node out of the operational range (calculated using the values in *pX, pY,* and *radius* data structures). The function **accepts a** message when: (i) its time stamp is inside of the interval in the *timeWindow* and (ii) it was sent by a node inside of the operational range (calculated using the values in *pX, pY,* and *radius*).

4.2.2 CIFH Operation

CIFH's first phase is the setup phase. It starts by emptying the *WVPS* data structure and the *R* data structure (line 1 in Fig. 8). After this, the setup phase creates a time interval and stores it in the *timeWindow* data structure. The time interval in the *timeWindow* data structure is created by adding to the current time (known through the *currentTime*() function) a tolerance interval (which is stored in the *TInt* data structure and is set according to a predefined acceptable time tolerance) (line 2 in Fig. 8). The node position on x-axis and y-axis is stored in the data structures *pX* and *pY*, respectively (such positions are known through the function *getNodePosition*() (line 3). The node operating range is set in radius (the operating range is known through the function *getNodeRadius*()) (line 4).

The periodical phase starts after the setup phase. During this phase, the algorithm receives messages from the other nodes executing LIFH over the network and stores it in the *HFPS* data structure (line 5 in Fig. 8). The algorithm receives messages generated within the interval in the *timeWindow* data structure and within the operational range of the node (calculated using the values in *pX*, *pY,* and radius data structures) (line 6). The identification of the node that sent the received message is stored in the *FN.id* data structure.

If the *WVPS* data structure is still empty (line 7 in Fig. 8), the algorithm stores all the phenomena within the *HFPS* data structure into the *WVPS* data structure (line 8) and marks all the recently stored phenomena in *WVPS* data structure with the identification of the *FN.id* data structure (line 9).

Else, if the *WVPS* data structure is not empty (line 10 in Fig. 8), for each phenomenon in the *HFPS* data structure (line 11), it empties the differences dictionary (*Dd*) (line 12), and for each phenomenon *P'* stored in the *WVPS* data structure (line 13), it inserts the absolute value of the difference between |*P* − *P'*|, with the pair (*P*, *P'*) as key (line 14). After adding all the pairs (*P*, *P'*) in *Dd* dictionary, it verifies all

Input: Multiple Hephaestus Feature Maps (HFPS);
Output: An Fusion Phenomena Map (WVPS);

Set-Up Phase:

```
1.  Fusion Phenomena Map: WVPS = ∅, R = 0;
2.  timeWindow = [ currentTime() - Δt , currentTime() + Δt ];
3.  [pX, pY] = getNodePosition();
4.  radius = getNodeRadius();
```

Periodical Phase:

```
5.  While true:
6.     HFPS = Receive_Hephaestus_Map(timeWindow, pX, pY, radius));
7.     If WVPS == ∅:
8.        Store all phenomena in HFPS on WVPS ;
9.        Mark all phenomena in WVPS with the FN.id;
10.    Else:
11.       For each phenomenon P on HFPS :
12.         Differences Dictionary Dd = ∅ ;
13.         For each phenomenon P' on WVPS :
14.             Dd.insert ( (P,P') ,  |P -  P'| );
15.         End For each;
16.         For each key (P, P') on Dd :
17.            Select the lower  |P -  P'| :
18.               If the lower |P -  P'| < Tu :
19.                  Phenomenon P' = (P + P') / 2;
20.                  Update P' on WVPS;
21.                  Mark P' with the FN.id;
22.               Else ( If the lower |P -  P'| > Tu ) :
23.                  Store P on WVPS;
24.                  Mark P with the FN.id;
25.         End For each;
26.      End For each;
27.   If ( Rounds == R):
28.      Send the WVPS to the BS.
29.   End If.
30. Continue While.
```

Fig. 8 CIFH

entries in *Dd* (line 16). The lower difference $|P - P'|$ represents the closer pair (P, P') (line 17 in Fig. 8).

Different units (temperature, humidity, accelerometer, etc.) have different threshold units (*Tu*), which means that despite closer values (P, P') , if the difference is greater than the threshold stored into the *Tu* data structure, they represent different phenomena.

So, if the lower difference $|P - P'|$ is lower than the value in *Tu* data structure (line 18 in Fig. 8), the phenomenon P' is now the mean $P' = (P + P')/2$ (line 19) and it is updated in *WVPS* (line 20). Also, this phenomenon P' is marked with the identification of the FN that recognized it (in *FN.id* data structure) (line 21 in Fig. 8). Else, if the lower difference $|P - P'|$ is greater than the value in *Tu* data structure (line 22), it represents a different phenomenon and is stored into the *WVPS* data structure (line 23) and also is marked with the id of the FN that recognized it (in *FN.id* data structure) (line 24). Then, to conclude, after a certain number of rounds, which are stored in *R* data structure (line 27 in Fig. 8), the content of the *WVPS* data structure is sent to the sink node (line 28).

In the sink node, any application can use the information provided by Hephaestus to be aware of the phenomena occurring in the monitored area and make decisions. Since the information produced by Hephaestus is generated without being aware of the specific requirements (as data interval, rates, and states) of any application and can be used for any application that demands it, Hephaestus is an information fusion algorithm for multiple applications in WSNs that does not need to be aware of the requirement of the deployed applications.

5 Evaluation

In this section, we describe the experiments performed for assessing Hephaestus regarding different aspects. We defined two main goals for the evaluations. **The first goal** is to analyze the overhead of Hephaestus in terms of communication overhead and energy consumption of WSNs. **The second goal** is to analyze the accuracy of Hephaestus in terms of its success in correctly inferring the states of the applications used in our case study.

To assess the overhead of Hephaestus in terms of communication and energy consumption of WSNs, we compare: (i) the moving average filter (MAF), a well-known data fusion algorithm; (ii) LIFH used as an input for MAF; and finally, (iii) the full Hephaestus algorithm (LIFH and CIFH) used as input to MAF. To assess Hephaestus accuracy (in terms of success in correctly inferring the states of the applications), we compare the accuracy when using the moving average filter (MAF) for multiple applications, but without being aware of the requirements of the applications, and using Hephaestus as input to MAF, also without being aware of the requirements of the applications.

In the following subsections, we describe the environment configuration, the application scenario, and the metrics used in our experiments. Thereafter, we discuss the results obtained in the performed experiments.

5.1 Environment Configuration and Application Scenario

In the performed simulations, we used the SUNSPOT manager tool Solarium since it enables managing virtual spots alongside real SUNSPOT sensor nodes. We used Solarium manager tool with all its default configurations. We used the default Solarium LQRP (Link Quality Routing Protocol) [29] as routing protocol to transmit data among sensor nodes.

In all conducted experiments, a WSN composed of 22 SUNSPOT sensor nodes was considered. The nodes were arranged in a flat and static network topology. The nodes have the same sensing units, all sensor nodes sense temperature samples, but they can have different sensing rates. All experiments were performed in an environment consisting in a 50 m $\times$ 50 m field. The network nodes were randomly arranged over a squared area defined as $\{(0, 0), (50, 0), (0, 50), (50, 50)\}$. To verify the impact of monitoring and transmitting over a long-range area, the sink is located far from any sensor node, at coordinates (200, 100). All sensor nodes start with 0.5 joules as initial energy within their batteries [42]. In the simulated network, 16 sensors are collector nodes (CNs), 5 are fusion nodes (FNs), and 1 is a sink node (SN). CNs are responsible for collecting data, and the FNs are responsible for executing LIFH and CIFH procedures. We have a set of 4 sensor nodes performing LIFH and 1 sensor node performing CIFH. A sensor node cannot act as both fusion node and collector node simultaneously.

The experiments to assess Hephaestus accuracy were performed in the context of a scenario for Smart Grid applications. Two applications from the Smart Grid domain were created and deployed in WSNs to share the available communication and sensing infrastructure. The first one was an application for overhead power line monitoring (OPLM) and the second an application for battery monitoring (BM) used in the transmission towers. A transmission tower is a tower used to support an overhead power line. An overhead power line is an electric power transmission line suspended by towers. The overhead power lines are the most common means to transmit energy. In the Smart Grid context, the transmission tower is also responsible for storing energy using local batteries. Both applications share the same kind of information (temperature), but with different data ranges. Moreover, they are correlated, in the sense that the behavior of an application can affect the behavior of the other. For instance, at 75 °C, there could be damages to the overhead power line. If the line is damaged, there will be no energy supply through the overhead power lines. Consequently, there is no power to be stored by the battery, even though 75 °C represents a normal temperature for the BM application. Thus, 75 °C represents a normality situation to the BM application (the usual battery temperature is around 40–144 °C), but the same temperature means that the line could become damaged

Table 2 Cases considered in the scenario

Cases	Temperature range
C1	40–65 °C on the overhead power line; 40–144 °C on the battery
C2	65–75 °C on the overhead power line; 40–144 °C on the battery
C3	40–65 °C on the overhead power line; Over 144 °C on the battery
C4	65–75 °C on the overhead power line; Over 144 °C on the battery

for the OPLM application. (The usual and safe overhead power line temperature is around 40–65 °C.) Battery and overhead power line temperatures changed according to the thermal models presented in [37].

To represent how Hephaestus infers environment features to describe each data range considered by the applications, we set four cases called C1, C2, C3, and C4. Each case represents a specific state of the applications along time, a given event to be detected. C1 represents ideal conditions for both applications (a safe condition, where there is no need to perform preventive actions to avoid damage of the power line and of the battery). C2 represents an increase in the overhead power line temperature. C3 represents an increase in the battery temperature. C4 represents an increase in the temperature of both the battery and the overhead power line. The data ranges for the time slots are summarized in Table 2. During C1, the overhead power line is in normal condition, it generates temperature samples from 40 to 65 °C, and the battery is in normal condition and generates from 40 to 144 °C. During C2, the overhead power line is overloaded and generates temperature samples from 65 to 75 °C, while the battery is in normal condition and generates from 40 to 144 °C. In C3, the overhead power line is in normal condition and generates temperature samples from 40 to 65 °C, while the battery is overloaded and generates samples over 144 °C. Finally, in C4, the overhead power line is overloaded and generates temperature samples from 65 to 75 °C, while the battery is also overloaded and generates samples over 144 °C.

5.2 Metrics

The following metrics were used in the performed experiments for evaluating the overhead of Hephaestus in terms of consumption of communication and energy resources of WSNs: (i) the number of messages transmitted by the nodes running in Hephaestus, (ii) the size of the transmitted messages by the nodes running in Hephaestus, (iii) the percentage of free bytes in RAM (BR) and the percentage of free bytes in flash memory (BF), and (iv) WSN lifetime (WL).

The *number of messages transmitted* is defined as the sum of the messages transmitted by the nodes running in Hephaestus to the sink node, in average, during a given period of time. We set the period of time as the duration of a single experiment. The *size of the transmitted messages* is defined as the length in bytes of the messages transmitted by node, in average, during a period of time. The *BR metric* is defined as the ratio between (i) the difference of the total amount of RAM in the SUNSPOT platform and the amount of memory used by the fusion procedures installed in the nodes and (ii) the total amount of RAM available in the SUNSPOT platform. The *BF metric* is defined similarly, but for the Flash memory. The *WL metric* is defined as the time elapsed from the beginning of the algorithm execution until the moment in which the first node of WSNs dies, which is the time elapsed until the first node in WSNs has completely depleted its energy. We used this metric since when the first node dies, the network enters into an instability period [8]. To evaluate the energy consumed by the execution of Hephaestus, we used the energy model presented in the next section.

The following metrics were used in the performed experiments for evaluating the accuracy of Hephaestus in terms of "behavior's change detection rate" [19]. We measured the "behavior's change detection rate" using false positives (FP), false negatives (FN), true positives (TP), and true negatives (TN). Each application has a set of states, which describes its behavior. According to our scenario, such states could be battery damaged, battery healthy, power line damaged, and power line healthy. FP indicate an incorrect detection of application state change (for example, an OPLM application that detects a transmission failure when there is no such a failure), and FN indicate a missing detection of application state change (an OPLM application that does not detect a transmission failure when the temperature reaches a given threshold that denotes a failure). TP indicate that an application has changed states correctly (a transmission failure is detected and it really occurred), and TN indicate that an application has maintained its state correctly. True occurrences are defined as the sum of TP and TN.

5.3 Energy Model

In this subsection, we present the energy model used to evaluate Hephaestus' energy consumption [26]. Sensor nodes in general have three functional modules: communication, sensing, and processing. The state of a sensor node can be described by the combined states of all its individual modules. We assume that each functional module has only two states, namely active and inactive. Each module can only stay in one of the two states, and its status does not affect other functional modules. The sensing and communication modules are the major functional parts we are concerned with in our proposal. Thus, the energy consumption of a sensor node is a function of the node state and the time during which the node remains in such state. Whenever the states are determined, a sensor node consumes a fixed energy rate during that time period. When the state of the corresponding functional module of a sensor node is

active, the energy consumption is much greater than when it is inactive. Assuming that an insignificant amount of energy is consumed when the corresponding function module remains inactive, we consider that the energy consumption in that state is 0. Overall, the energy consumption of a sensor node during time t is calculated by $E(T) = E_c + E_s$, where E_c represents the energy consumption of communication module and E_s represents the energy consumption of sensing module.

For the ease of analysis, we assume that the data exchange between two neighboring sensor nodes (within one-hop communication range) belonging to the same WSN is done through direct communication. The energy consumption of transmitting $l - bit$ (message of l bits of size) data over distance d is defined as $E_{tx}(l, d)$ [17]:

$$E_{tx}(l, d) = E_{elec} \times l + \varepsilon_{amp} \times l \times d^2 \tag{4}$$

where E_{elec} and ε_{amp} are hardware-related parameters [17]. We also assume that the receiver does not consume energy in the data exchange process. For any two-distance sensors (outside one-hop communication range, but still belonging to the same WSN), the data communication is carried out by using shortest path-based multihop routing protocol (please note that the routing process is out of scope of our work). The energy consumption of transmitting $l - bit$ (message of l bits of size) data from source (src) to the destination (des) is defined as $E_{tx}(l, src, des)$:

$$E_{tx}(l, src, des) = \sum_{i=1}^{k} E_{tx}(l, d) \tag{5}$$

where d is replaced from hop to hop, i is an iterator, and k is the minimum hop count the data travels from source to destination and $E_{tx}(l, d)$ remains the same meaning as shown in Eq. (4). The energy consumption of sensing module is calculated by a linear equation [17]:

$$E_{s_i} = ER_{s_i} \times t_{s_i} \tag{6}$$

where ER_{s_i} represents energy consumption of service i in one time unit and t_{s_i} represents the period of time for performing service i.

5.4 Evaluating Hephaestus Overhead

This section describes the experiments performed to evaluate Hephaestus. For this set of experiments, instead of using the Smart Grid applications (described in Sect. 5.1), we created synthetic applications with varying requirements. We simulated 2, 4, 6, 8, and 10 applications running simultaneously in the network. For each application, we randomly assigned one sensing unit from a set of 5 different sensing

units (accelerometers, temperature, light, humidity, and presence). For the assigned sensing unit, we randomly assigned sensing rates varying from 1 to 5 s, using the procedures explained in [33]. It is discussed in the literature that random monitoring tasks may not always represent real applications; however, the diversity they provide is sufficient for this group of experiments as explained in [33]. Each application had an interval randomly generated varying from 0 to 1000 (changing the measure unit according to the sensing type).

The nodes of WSNs started with 0.5 J of initial energy. Each experiment was executed for 30 min, and the amount of energy consumed in each one was assessed. Each experiment was repeated 30 times, obtaining a confidence interval of 95%. LIFH was set up its sample size as 200 samples (LIFH waits until it receives 200 samples to start its procedures). CIFH was set to execute 4 rounds (one round for each one of the 4 fusion nodes sending the results for LIFH).

5.4.1 Experiment for Assessing the Number of Transmitted Messages

The results of the experiments for evaluating the number of transmitted messages are shown in Fig. 9. From the figure, we observe that the number of messages transmitted when using only MAF for 2, 4, 8, and 10 applications is greater than the number of messages transmitted when using Hephaestus (LIFH and CIFH) or just LIFH. Hephaestus sends a single message, regardless of the number of applications in the network. In contrast, one instance of MAF is executed for each application, which increases the number of messages transmitted as more applications are executed in the network. In Fig. 9, we can observe that the number of messages transmitted when using Hephaestus (LIFH and CIFH) was lower than using only LIFH. CIFH is responsible for decreasing the number of messages transmitted from FN to the sink node. When using both CIFH and LIFH (Hephaestus) with MAF, the number of transmitted messages to the sink node decreases in relation to the number of transmitted messages using only LIFH.

5.4.2 Experiment for Assessing the Size of the Transmitted Messages

Figure 10 shows the results of the experiments for evaluating the size of the transmitted messages. We can observe that the size of the transmitted messages when using only MAF for 2, 4, 8, and 10 applications remains constant regardless of the number of applications in the network. We can also observe that when using (i) Hephaestus (LIFH and CIFH) or (ii) only LIFH, the size of the transmitted messages increases almost as linearly as more applications are installed in the network. The size of the transmitted messages when using Hephaestus (LIFH and CIFH) presents a larger size when compared to the size of the transmitted messages when using just LIFH, since CIFH stores each different inferred phenomenon with an identification of the node that identified such phenomenon through LIFH. The size of the messages sent using Hephaestus is greater than the messages sent using only LIFH due to the

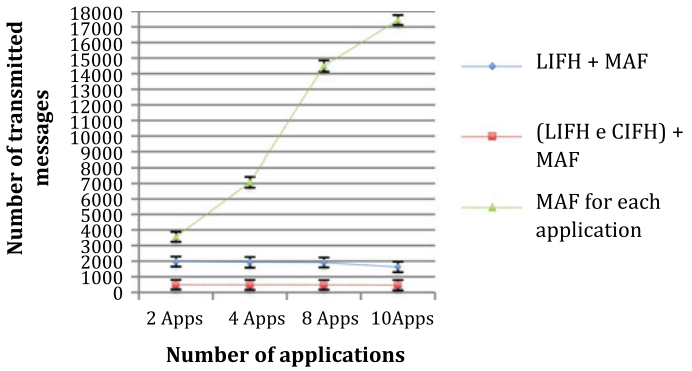


Fig. 9 Number of transmitted messages

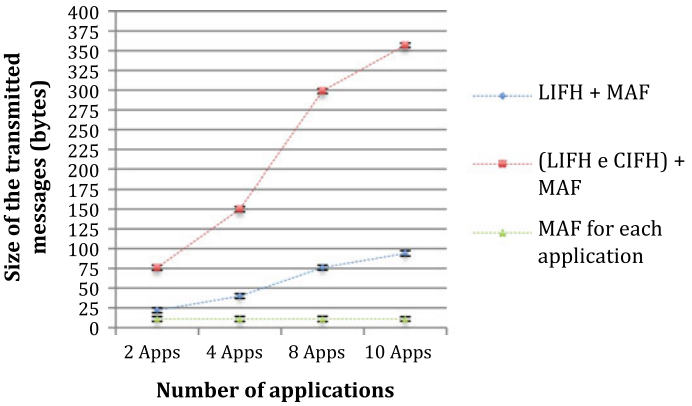


Fig. 10 Size of transmitted messages

need of storing an identifier for each inferred phenomenon. Additionally, as more phenomena are inferred, the size of the messages increases.

5.4.3 Experiment for Assessing the Network Lifetime (WL)

Figure 11 shows the results for assessing the network lifetime. We can observe that the values of the network lifetime (for 2, 4, 6, 8, and 10 applications) using (i) Hephaestus (LIFH and CIFH) or (ii) only LIFH are greater than the lifetime using only MAF. We can also notice that the network lifetime when using Hephaestus is greater than using LIFH for 2, 4, 6, and 8 applications. However, when using 10 applications, the network lifetime is greater using only LIFH. There are two major facts that explain the obtained results for the network lifetime: (i) the experiments for assessing the number of transmitted messages showed that such value obtained by

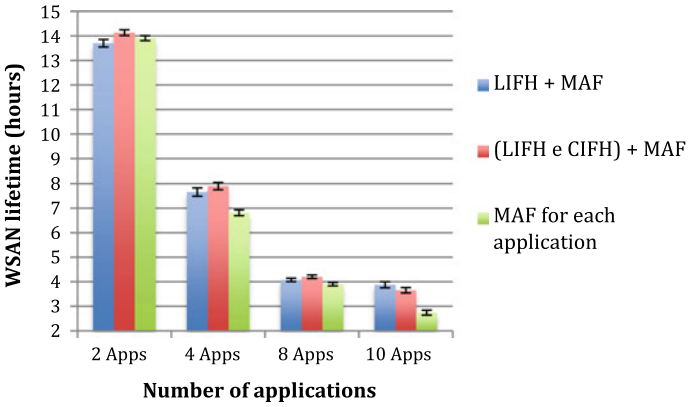


Fig. 11 WSN lifetime

using Hephaestus does not increase as more applications are installed in the network (Fig. 9); and (ii) the experiments for assessing the size of the transmitted messages showed that such size when using Hephaestus increases as more applications are installed in the network (Fig. 10). These two facts depict a trade-off between number of transmitted messages and message size. The network lasts longer when using only LIFH for 2, 4, 6, and 8 applications, since although LIFH sends more messages than Hephaestus, the size of the messages transmitted by LIFH is smaller than the message size in Hephaestus.

5.4.4 Experiment for Assessing the Memory Consumption

This subsection describes the experiment performed to assess Hephaestus in terms of memory usage. We evaluate Hephaestus in terms of *the percentage of free bytes in flash memory* and in terms of *the percentage of free bytes in RAM*. To achieve the first value, we installed Hephaestus in the flash memory of a sensor node and checked the amount of flash memory consumed. To assess Hephaestus in terms of *the percentage of free bytes in RAM*, we executed Hephaestus in the scenarios for 2, 4, 8, and 10 applications and checked the amount of RAM memory consumed while Hephaestus was running in a sensor node. It is important to notice that we used SUNSPOT sensor nodes in this experiment. A SUNSPOT sensor node has 1 MB of RAM and 4 MB of flash memory.

The percentage of bytes in RAM (BR) is depicted in Fig. 12. The percentage of free bytes in flash memory (BF) was 99.60% for Hephaestus and 99.90% for MAF. Concerning the value of BR, MAF presented lower memory consumption than Hephaestus for any number of applications. This result was already expected since MAF is a lightweight data fusion algorithm [19]. Despite being lightweight, MAF executes an average for each application, regardless of the requirements of the

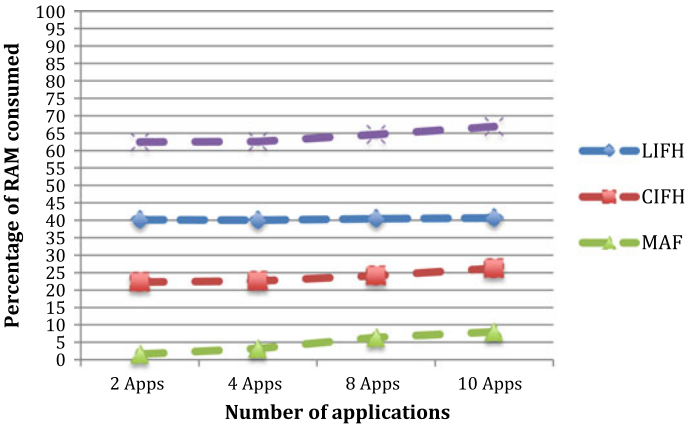


Fig. 12 Percentage of RAM consumed

applications. This behavior reflects on the accuracy of MAF (which will be discussed in the experiments for evaluating the accuracy of Hephaestus) and produces a huge number of messages in the network, which decreases the network lifetime when using MAF.

LIFH presented higher memory consumption than CIFH. This behavior of Hephaestus is due to the need of LIFH to store the samples from the environment in memory, while CIFH just processes the information from LIFH. As the number of samples in the experiments remained the same, LIFH did not present a significant increase in memory consumption; its memory consumption remained almost linear, regardless of the number of applications. On the other hand, CIFH slightly increased its memory consumption. Indeed, CIFH is more impacted by the processing and storing of more information, as it does not store the samples from the environment in its RAM memory. Considering Hephaestus (LIFH + CIFH), our experiment showed that LIFH + CIFH consumes 66% of the RAM, while MAF consumes 8%. This result can be explained by the fact that Hephaestus (LIFH + CIFH) is more complex than MAF.

The size of the Hephaestus prototype in the SUNSPOT RAM memory was of 641.362 KB with 17.967 KB of standard deviation. It is important to notice that the SUNSPOT protocol stack contains the application layer and three middle layers (the *edemo_device*, *multihop,* and the *espot_device* libraries), which may increase the total size of the application footprint when compared to other platforms (e.g., TelosB) [10].

Table 3 Experiments to assess accuracy

	Hephaestus and MAF (BM, OPLM)	MAF (BM)	MAF (OPLM)	MAF (BM, OPLM)
T1	99.66% ± 0.001176063	100%	100%	72.1% ± 4.9
T2	99.26% ± 0.0023	95.2% ± 2.4	93.8% ± 1.4	77.4% ± 2.9
T3	95.88% ± 0.011095	94.3% ± 2.1	93.4% ± 1.5	44% ± 6.1
T4	84.64% ± 0.018099	94.2% ± 0.3	94% ± 1.2	78% ± 2.1

5.5 Evaluating Hephaestus Accuracy

This section describes the simulated experiments performed to assess the accuracy of Hephaestus and MAF (moving average filter) together and separately in terms of their ability to correctly identify the different events of interest for applications described in our case study. We performed four experiments. In the first simulated experiment, we used Hephaestus and MAF together. In the second (denoted as BM to indicate the battery monitoring application was executed) and third experiments (execution of the OPLM application), we applied MAF in the received data of each application separately. Finally, in the fourth experiment, we used only MAF (for both BM and OPLM applications). In the second (BM) and third experiments (OPLM), the algorithms need to be aware of the requirements of the deployed applications. In the first and fourth simulated experiments, they did not need to be aware of such requirements. Hephaestus and MAF sample sizes were set as 200 samples. For evaluating the accuracy of Hephaestus, the experiments lasted 1 h and were repeated 30 times, obtaining a confidence interval of 95%.

During these experiments, there were four time slots (T1, T2, T3, and T4). So, for each time slot, Hephaestus and MAF were tested for comparing the achieved results in terms of true occurrences (TP + TN). In each time slot, we intended to reproduce the behavior of the cases presented in our case study. In this regard, T1 intends to reproduce the behavior in case 1 (C1), T2 to represent C2, and so on.

Table 3 shows the achieved results of the comparison between Hephaestus and MAF in terms of accuracy. The first row represents how effective Hephaestus is in detecting changes in the application's behavior. This row denotes values achieved when Hephaestus deals with multiple applications simultaneously. The second row represents how effective MAF is in detecting a change in the BM application behavior. The third row represents how effectively MAF is in detecting a change in the OPLM application behavior. The fourth row represents situations where MAF deals with multiple applications simultaneously (BM and then OPLM).

For the T1 time slot, both applications are in an ideal condition. In this case, given the unequal data ranges, the composition of both applications generated a highly skewed dataset. Hephaestus was capable of correctly dividing the dataset

into two distinct groups, identifying the different datasets, and applying MAF in each set. The traditional MAF, on the other hand, grouped all the data into a single group. Applying MAF directly to the dataset biased the fusion result. Such bias is responsible for the worst accuracy of MAF (72.1%) in comparison with Hephaestus' accuracy (99.66%).

For the T2 time slot, both applications overlapped, which produced a moderate skewed dataset. Hephaestus recognized this dataset as a single dataset, but with a slightly shifted mean over the more concentrated data range. The behavior of this mass of data indicates a concentration of data generated over the overhead power line , which is overloaded. The shifted mean of data generated over the overhead power line was responsible for the greater accuracy of Hephaestus (99.26%) over the MAF (77.4%), which produced a non-representative data fusion for both applications more frequently than Hephaestus does.

For the T3 time slot, a highly skewed multimodal dataset was formed by the composition of the different applications. Due to the high skewness of the dataset, Hephaestus was able to identify that there were different applications in the monitored area. Different from Hephaestus, MAF equally weighted all data points and executed an average that does not represent neither application. The difference between the accuracy achieved by Hephaestus (95.88%) in comparison with MAF's accuracy (44%) is due to the biased fusion result of MAF.

By comparing the accuracy of Hephaestus on T4 time slot with the accuracy on other time slots, we can notice a fall in the result. The dataset formed on T4 time slot is an almost symmetric multimodal dataset. Due to its symmetry, depending on the samples Hephaestus receives as input, the dataset is recognized as a symmetric dataset, which does not need to be divided, producing a single dataset. As T4 presents an almost symmetric dataset, the accuracy of Hephaestus is lower (84.64%), than its accuracy in the other time slots (99.66% for T1, 99.26% for T2, and 95.88% for T3). However, compared with MAF, Hephaestus has a better accuracy result (78% for **MAF**).

5.5.1 Effect of the Sample Size on T4

Considering that the accuracy achieved by Hephaestus in T4 time slot was lower (84.64%) than the accuracy for the other time slots (99.66% for T1, 99.26% for T2, and 95.88% for T3), we decided to further investigate how much the sample size impacts the Hephaestus' accuracy. For this experiment, we put Hephaestus to run on T4 time slot analyzing 200, 400, 600, and 800 samples. All experiments were repeated 30 times, and the results had a 95% confidence interval. The results in Table 4 reflect that as Hephaestus analyzes more samples, its accuracy increases. This result is expected, since as the sample size increases, more complete is the observation of the real world and lower the associated uncertainty. This result is relevant since it presents a solution for a situation where Hephaestus is unable to correctly recognize a dataset due to a particular entropic configuration.

Table 4 Sample size versus accuracy

Sample size	Accuracy on T4
200	84.64% ± 0.018099
400	95.07% ± 0.007517
600	97.33% ± 0.004674
800	99.11% ± 0.002032

5.6 Analysis of Results

In this section, we have presented the experiments conducted with Hephaestus regarding two established goals. The **first goal** was to analyze Hephaestus for the purpose of evaluating its overhead in terms of communication and energy resources of WSNs. The **second goal** was to analyze the accuracy of Hephaestus in terms of successes in correctly inferring the states of the applications in a Smart Grid application scenario, without the knowledge of specific application requirements (for instance, data intervals or the potential events of interest).

The results achieved by the experiments showed that Hephaestus does not increase the network traffic as more applications are deployed and/or more phenomena are taking place in the network. However, a trade-off was identified regarding the increase of the message size produced by Hephaestus. As more applications are deployed and/or more phenomena are taking place in the network, the largest the size of the messages produced by Hephaestus. The trade-off between message traffic and message size is reflected in the WSN lifetime experiment. The WSN lifetime increases when using Hephaestus, in relation to MAF. The message traffic of MAF is greater than the message traffic of Hephaestus, as MAF is executed for each application individually. Hephaestus presented higher values of accuracy than using only MAF for the Smart Grid application scenario. This result was due to the information produced by Hephaestus, even without being aware of the requirements of the deployed application.

6 Final Remarks

Hephaestus was proposed as an entropic information fusion algorithm, which uses mean, kurtosis, and skewness to apply a heuristics that infers from dataset multiple features, for multiple applications. Each inferred feature relates to a phenomenon occurring in the monitored area. We consider a phenomenon as any situation in the monitored area (e.g., an overheating in a certain part of the area) that generates data in a specific range that may or may not have correlation to an event of interest for an application in the real world. As more different phenomena generate data in the monitored area, more difficult is to separate and infer about the different phenomena individually, in other words, more difficult to identify the different events of interest

for applications running in WSNs. In such case, as more phenomena generate data in the monitored area, more complex is the data entropy.

This chapter presents the detailed description of Hephaestus along with several experiments to evaluate its performance and overhead. The current drawbacks presented by Hephaestus are the increase in the message size, as more applications are installed in the network, which impacts negatively on the system lifetime.

Acknowledgements This work is partly supported by the following Brazilian funding agencies: National Council for Scientific and Technological Development (CNPq), Financier of Studies and Projects (FINEP), and the Foundation for Research of the State of Rio de Janeiro (FAPERJ). Luci Pirmez, Flavia C. Delicato, and Paulo F. Pires are CNPq Fellows.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Comput. Netw. **38**(4), 393–422 (2002)
2. Aquino, G., Pirmez, L., Farias, C.M., Delicato, F.C., Pires, P.F.: HEPHAESTUS: a multisensor data fusion algorithm for multiple applications on wireless sensor networks. In: Information Fusion (FUSION), 2016 19th International Conference on Information Fusion, July 2016
3. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 28, Pages 2787–2805. ISSN 1389-1286. Oct 2010
4. Bayes, M.R., Price, M.: An essay towards solving a problem in the doctrine of chances. In: Bayes, F.R.S. Communicated by Price, in a Letter to John Canton, A.M.F.R.S.: Bayes, M.: Free Download and Streaming: Internet Archive. Philosophical Transactions, 1763
5. Bicocchi, N., Mamei, M., Zambonelli, F.: Self-organizing virtual macro sensors. ACM Trans. Auto. Adapt. Syst. **7**(1), Article 2, May 2012
6. Bird, C.M., Beers, T.C.: Astronomical applications of distribution shape estimators. Astro. J. **105**, 1596–1606 (1993)
7. Cauteruccio, F., Fortino, G., Guerrieri, A., Terracina, G.: Discovery of hidden correlations between heterogeneous wireless sensor data streams. In: International Conference on Internet and Distributed Computing Systems, pp. 383–395. Springer International Publishing, Sept 2014
8. Costa, E.A. et al.: CAMAW: a clustering algorithm for multiple applications in WSAN. In: International Conference on Systems and Networks Communications, vol. 10, pp. 81–89. Barcelona, Nov 2015
9. Dasarathy, B.V.: Sensor fusion potential exploitation-innovative architectures and illustrative applications. Proc. IEEE **85**(1), 24–38 (1997)
10. Datasheet, T.: Crossbow Inc. http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf (2013)
11. Degroot, M.H., Schervish, M.J.: Probability and Statistics. Addison-Wesley (2002)
12. Delicato, F.C., Pires, P.F., Pirmez, L., Carmo, L.F.: A service approach for architecting application independent wireless sensor networks. Cluster Comput. **8**(2–3), 211–221 (2005)
13. Delicato, F.C., Pires, P.F., Pirmez, L., da Costa Carmo, L.F.R.: A Flexible Web Service Based Architecture for Wireless Sensor Networks, IEEE Computer Society (2004)
14. Delicato, F.C., Protti, F., Rezende, J.F., Pirmez, L.: An efficient heuristic for selecting active nodes in wireless sensor networks. Comput. Netw. **50**, 3701–3720 (2006)
15. Efstratiou, C.: Challenges in Supporting Federation of Sensor Networks (2010)
16. Efstratiou, C., Leontiadis, I., Mascolo, C., Crowcroft, J.: A shared sensor network infrastructure. In: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems—SenSys '10, p. 367. ACM Press, New York, New York, USA (2010)

17. Farias, C.M.: A framework for developing smart space applications using shared sensor networks. D. Sc. Dissertation. Federal University of Rio de Janeiro, Brazil (2014)
18. Farias, C., Pirmez, L., Delicato, F., Carmo, L., Wei Li Zomaya, A.Y., DE Souza, J.N.: Multisensor data fusion in shared sensor and actuator networks. In: Information fusion (FUSION), 2014 17th International Conference on Information Fusion, July 2014
19. Farias, C.M., Pirmez, L., Delicato, F.C., Dos Santos, I.L., Zomaya, A.Y.: Information fusion techniques applied to shared sensor and actuator networks. In: Local Computer Networks (LCN), 2012 IEEE 37th Conference on Local Computer Networks, Oct 2012
20. Goos, P., Meintrup, D.: Statistics with JMP: Graphs, Descriptive Statistics and Probability, pp. 368. Wiley, 2015
21. Gungor, V.C., Lu, B., Hancke, G.P.: Opportunities and challenges of wireless sensor networks in smart grid—a case study of link quality assessments in power distribution systems. System (2010)
22. Joanes, D.N., Gill, C.A.: Comparing measures of sample skewness and kurtosis. J. Royal Stat. Soc. Series D (The Statistician) (1998)
23. Khaleghi, B., Khamis, A., Karray, F.O., Razavi, S.N.: Multisensor data fusion: A review of the state-of-the-art. Inf. Fusion **14**(1), 28–44 (2013). https://doi.org/10.1016/j.inffus.2011.08.001
24. Labs, S.: Sun™ SPOT Programmer's Manual. Communication (2010)
25. Leontiadis, I., Efstratiou, C., Mascolo, C., Crowcrof, J.T.: SenShare: transforming sensor networks into multi-application sensing infrastructures. In: Picco, G.P., Heinzelman, W. (eds.), Proceedings of the 9th European Conference on Wireless Sensor Networks (EWSN '12), pp. 65–81. Springer, Berlin, Heidelberg (2012)
26. Li W., Delicato, F.C., Pires, P.F., Lee, Y.C., Zomaya, A.Y., Miceli, C., Pirmez, L.: Efficient allocation of resources in multiple heterogeneous wireless sensor networks. J. Parallel Distrib. Comput. **74**(1), 1775–1788 (2014)
27. Mus, P., Kr, P.: E-BACH: entropy-based clustering hierarchy for wireless sensor networks. In: 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), vol. 3, pp. 231–232. IEEE, Dec 2015
28. Nakamura, E.F., Loureiro, A.A.F., Frery, A.C.: Information fusion for wireless sensor networks: methods, models, and classifications. ACM Comput. Surv. (2007)
29. Paschoalino, C.R., Madeira, E.R.M.: A scalable link quality routing protocol for multi-radio wireless mesh networks. In: 2007 16th International Conference on Computer Communications and Networks, pp. 1053–1058 (2007)
30. Pearson, K.: Contributions to the mathematical theory of evolution. II. Skew variation in homogeneous material. Philos. Trans. Royal Soc. London (1895)
31. Perera, C., Zaslavsky, A., Christien, P., Georgakopoulos, D.: Context aware computing for the internet of things: a survey. IEEE Commun. Surv. Tutor. **16**(1), 414–454 (2014)
32. Pottie, G.J., Kaiser, W.J.: Wireless integrated network sensors. Commun. ACM, May 2000
33. Raghunathan, V., Schurgers, C., Park, S.P.S., Srivastava, M.B.: Energy-aware wireless microsensor networks. IEEE Signal Process. Mag. **19**(2), 40–50 (2002)
34. Rawat, P., Singh, K.D., Chaouchi, H., Bonnin, J.M.: Wireless sensor networks: a survey on recent developments and potential synergies. J. Supercomput. **68**(1), 1–48 (2013). https://doi.org/10.1007/s11227-013-1021-9
35. Safia, A.A., Al Aghbari, Z., Kamel, I.: Phenomena detection in mobile wireless sensor networks. J. Netw. Syst. Manage. **24**(1), 92–115 (2016)
36. Sanquetta, C.R., Behling, A., Dalla, C.A.P., Péllico, N.S., Rodrigues, A.L., Simon, A.A.: A model based on environmental factors for diameter distribution in black wattle in Brazil. PLoS ONE **9**(6), e100093 (2014)
37. Schläpfer, M., Mancarella, P.: Probabilistic modeling and simulation of transmission line temperatures under fluctuating power flows. IEEE Trans. Power Delivery **26**(4), 2235–2243 (2011)
38. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. (1948)
39. Silva, T.H., Vaz de Melo, P.O.S., Almeida, J.M.D., Loureiro, A.A.F.: Uncovering properties in participatory sensor networks. In: Proceedings of the 4th ACM International Workshop on Hot Topics in Planet-Scale Measurement—HotPlanet '12, pp. 33. ACM Press, New York, New York, USA (2012)

40. Singhal, D., Garimella, R.M.: Simple Median Based Information Fusion in Wireless Sensor Network. International Institute of Information Technology, Hyderabad, India—500 032 (2012)
41. SUNSPOT SDK, Release 6.0 of the. SUNSPOT SDK Release Notes—Sun™ SPOT Programmer's Manual Release v6.0 (Yellow)
42. Varalakshmi, L.M., Sudha, G.F., Jaikishan, G.: A selective encryption and energy efficient clustering scheme for video streaming in wireless sensor networks. Telecommun. Syst. **56**(3), 357–365 (2014)
43. Wagner, A., Speiser, S., Harth, A.: Semantic web technologies for a smart energy grid: requirements and challenges. In: Proceedings of 9th International Semantic Web (2010)
44. Wang, M., Perera, C., Jayaraman, P.P., Zhang, M., Strazdins, P., Ranjan, R.: City Data Fusion: Sensor Data Fusion in the Internet of Things. arXiv:1506.09118 (2015)
45. Watsham, T.J., Parramore, K.: Quantitative methods in finance. Cengage Learning EMEA (1997)
46. White, F.: Data fusion lexicon. San Diego, CA, U.S. Department of Defense, pp. 15 (Report ADA529661) (1991)
47. Whitmore, A., Agarwal, A., Xu, L.D.: The internet of things—a survey of topics and trends. Inf. Syst. Front. (2014)
48. Xu, Y., Saifullah, A., Chen, Y., Lu, C., Bhattacharya, S.: Near optimal multi-application allocation in shared sensor networks. In: Proceedings of the Eleventh ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 181–190. Chicago, Illinois, USA (2010)
49. Zhou, J., Lei, H., Feng, W., Hai-Han, L., Kai, Z.: An efficient multidimensional fusion algorithm for IoT data based on partitioning. Tsinghua Science and Technology (2013)
50. Zhu, H., Chen, S., Han, C., Lin, Y.: Fusion of possible biased local estimates in sensor network based on sensor selection. In: 2013 16th International Conference on Information Fusion (FUSION). Istanbul (2013)

Part VI
Topology Control and Routing

Underwater Networks for Ocean Monitoring: A New Challenge for Topology Control and Opportunistic Routing

Rodolfo W. L. Coutinho, Azzedine Boukerche and Antonio A. F. Loureiro

Abstract Underwater wireless sensor networks (UWSNs) have been proposed for ocean monitoring. In comparison with ocean monitoring technologies currently in use, UWSNs have the potential to revolutionize ocean monitoring applications by enabling (quasi) real-time data acquisition. However, the use of the acoustic channel as well as the characteristics of the aquatic environment present challenges in design of efficient networking protocols for underwater sensor networks. Due to the particular characteristics of UWSNs, well-established principles and designed networking protocols for terrestrial wireless sensor networks cannot be directly applied in underwater sensor networks. In this chapter, we discuss the peculiar characteristics of UWSNs, and how knowledge acquired over decades of research in terrestrial wireless sensor networks is impractical in underwater sensor networks. Moreover, we discuss intrinsic research challenges and provide some guidelines for the future design of topology control algorithms and opportunistic routing protocols for UWSNs, two main methodologies that can improve the performance of UWSN applications. In addition, we provide some future research directions toward enabling large-scale deployments of UWSNs for monitoring large areas of the ocean.

R. W. L. Coutinho (✉) · A. Boukerche
School of Electrical Engineering and Computer Science, 800 King Edward Ave. Ottawa, ON, Canada
e-mail: rodolfo.coutinho@uottawa.ca; rlimaco2@uottawa.ca

A. Boukerche
e-mail: boukerch@site.uottawa.ca

A. A. F. Loureiro
Federal University of Minas Gerais, Av. Antônio Carlos - 6627, Belo Horizonte, Minas Gerais, Brazil
e-mail: loureiro@dcc.ufmg.br

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_15

1 Introduction

Seventy percent of the Earth's surface is covered by ocean. The ocean plays an essential role for all living organisms on the Earth. This environment helps regulate weather and the temperature of our planet, absorbs 90% of heat from global warming, and more than 1/4 of human CO_2 emissions produced since 1800. It also produces 50% of the O_2 from photosynthesis. Moreover, the ocean provides primary resources and serves as a medium for commerce and transport of people and goods. These facts alone are sufficient to motivate the study of a deep understanding of the ocean and what lies beneath. Extensive knowledge about the ocean and its importance is fundamental for a sustainable and proper exploration, as well as for the protection of this environment.

Underwater wireless sensor networks (UWSNs) have been proposed for ocean monitoring missions. In this technology, a set of underwater sensor nodes is employed to monitor an area of interest and associated events. Each sensor node is responsible for collecting data in its surrounding area. Data delivery is then performed through multi-hop acoustic communication from underwater sensor nodes to surface sonobuoys (sinks) deployed at the surface of the ocean.

However, the use of the acoustic channel for underwater wireless communication incurs several daunting challenges. Underwater acoustic links suffer from low bandwidth, high and variable delay, high path loss and noise, shadow zones, multi-path propagation, and Doppler effect. Moreover, underwater acoustic communication is energy-hungry. Due to these characteristics of the underwater acoustic channel, it is impractical to UWSN the use of networking protocols designed for terrestrial wireless sensor networks, as well as topology control and data routing algorithms.

This chapter is devoted to discuss the challenges of topology control and opportunistic routing in large-scale UWSNs for ocean monitoring missions. It provides a detailed discussion regarding the intrinsic research challenges and recent advancements in the design of topology control and opportunistic routing protocols in UWSNs. By doing so, it provides a classification of topology control and opportunistic routing protocols in UWSNs, as well as some design guidelines that have emerged from the analysis of current protocols.

The remainder of this chapter is organized as follows. Section 2 introduces the general concepts of UWSNs. Section 3 describes some important characteristics of underwater sensor networks that poses challenges in the design of efficient networking protocols. Section 4 discusses various approaches for organizing the UWSN topology to improve the performance of networking protocols. Section 5 provides an in-depth discussion of the opportunistic routing paradigm in underwater sensor networks. Section 6 provides directions for future research in the topic. Finally, Sect. 7 concludes this chapter.

2 The Fundamental Role of Underwater Sensor Networks

Oceans are largely unknown, and their extensive monitoring as well as exploration activities in this environment are challenging. In fact, we know more about the moon and other planets than we do about our oceans. For instance, it was only in 1948 that scientists were able to prove the existence of life below 6,000 m. The crushing pressure and darkness of deep waters have made ocean the least explored areas of the planet [1]. Unfortunately, current realities regarding technologies free of human intervention for underwater events monitoring and control are not inspiring. Either sensor nodes without underwater communication capabilities (e.g., RAFOS instruments [2]) or wired-interconnected underwater sensor nodes have been used for underwater monitoring applications (e.g., Ocean Network Canada initiative [3]).

In addition, efficient ocean monitoring technologies are required for monitoring human offshore exploration among large areas of the ocean. This monitoring is required for avoiding situations resembling the biggest oil spill in the history of the USA, which happened on April 20, 2010, in the Gulf of Mexico, after an explosion on the British Petroleum (BP)-owned Deepwater Horizon drilling platform. To give an idea of the catastrophic nature of this event, the explosion on the platform killed 11 workers and released millions of barrels of oil in the Gulf of Mexico, impacting an area of 180,000 km^2 of ocean [4]. Therefore, the monitoring of exploration activities in the ocean is required, particularly now that oil/gas industries are moving their exploration sites from shallow water to the harsh environment of deepwater and ultra-deepwater fields.

Lately, considerable research efforts and achieved developments have enabled a new technology of underwater sensor networks. The current UWSNs are a particular kind of wireless ad hoc network and sensor network [5, 6], composed by underwater nodes and/or autonomous underwater vehicles (AUVs) with several sensor and actuator devices that are scattered by sonobuoys (sinks) deployed at the surface of the ocean. In this sensor network, underwater nodes are able to communicate with each other through multi-hop wireless acoustic communication. This capability allows them to collaboratively collect data from their surrounding environment and events of interest, and send the gathered data to a monitoring center.

In contrast to traditional approaches that use communication-less or wired architectures, the use of nodes with underwater wireless communication capabilities incurs several advantages for ocean monitoring missions. In the following, we discuss three main advantages [7]:

- *Instantaneous node failures detection.* In the traditional approach of the Argo project [8], an underwater node has a well-defined operating cycle, in which it spends most of its time drifting between several layers of depth to collect data; at the end of this cycle, it surfaces and off-loads the gathered data through satellite communication. Therefore, node failures will be inferred if the monitoring center does not obtain data from a node that is supposed to be off-loading its collected data. Conversely, in an underwater wireless sensor network, whenever a sensor or

an actuator in an underwater node is not working properly, the node could inform the monitoring center of its situation through multi-hop wireless communication. In addition, if a node experiences failure in its acoustic modem, this failure might be inferred from unusual silent time when the node should be transmitting.

- *Real-time data acquisition.* In the communication-less approach, data acquisition is obtained through satellite communication or at the end of the monitoring mission, when nodes are collected. In the first approach, data is obtained when each node surfaces and off-loads its gathered data. The inter-surface time of this approach can last for days. In the second approach, we might wait for years in order to obtain gathered data, which is the time of the underwater monitoring mission. The use of underwater wireless communication allows us to query the network and obtain collected data whenever it is necessary. Moreover, as soon as a node detects an event and starts collecting data, it can transmit this data to the monitoring center.
- *Online nodes reconfiguration.* In general deployments for ocean monitoring, nodes are programmed to perform their given task and remain doing that task during their lifetime. However, one might be interested in monitoring some events that occur seasonally (e.g., whales in a given area). Hence, it would be desirable to reconfigure the underwater nodes to support other applications when they are no longer needed in a prior application. In the traditional approach, a ship mission is needed to catch the nodes and, later on, to redeploy them with its novel setup. Conversely, the use of underwater wireless communication allows the monitoring center to send a command to each underwater node. Thus, it is possible to change the sampling rate of the sensors, the physical location of the node (horizontal and vertical movement), and the inclusion or removal of sensing tasks of physical variables (e.g., temperature, pressure, salinity, and pH).

The advantages mentioned above motivate the research toward large-scale deployments of UWSNs. Large-scale UWSN deployments are expected to change the current reality where no more than 5% of the volume of the ocean has been seen by the human eyes [9]. UWSNs are envisioned to make possible a new era in scientific and industrial underwater monitoring and exploration applications, such as the monitoring of marine life, pollutant content, geological processes on the ocean floor, oil fields, climate, and tsunamis and seaquakes and to collect oceanographic data, ocean and offshore sampling, navigation assistance, and mine recognition, in addition to being used for tactical surveillance applications. These applications will help fill the gap of our knowledge regarding the ocean and aquatic environments in general.

3 Characteristics of Underwater Sensor Networks

Underwater sensor networks have many peculiar characteristics that distinguish them from the well-investigated terrestrial wireless sensor networks (WSNs). In this section, we discuss some of the characteristics that directly affect the performance of a UWSN. We present a traditional UWSN architecture (Sect. 3.1) and discuss two

major factors increasing the deployment cost of a UWSN (Sect. 3.2). Moreover, we highlight the involuntary mobility of underwater nodes (Sect. 3.3) and review the underwater acoustic communication channel (Sect. 3.4) that are completely different from the counterpart terrestrial WSN.

3.1 Network Architecture

Traditionally, a UWSN architecture will consist of sensor nodes and underwater autonomous vehicles (UAVs) deployed underwater, as shown in Fig. 1. These devices will be provided with sensing, processing, storage, and underwater wireless communication capabilities. They will be scattered by sonobuoys (sinks) deployed at the surface of the ocean. The sensor nodes collect the data from the area and surrounding events of interest, and they send the gathered data to the surface sinks through multi-hop underwater acoustic communication. The surface sonobuoys are equipped with both underwater acoustic and radio frequency modems. Underwater acoustic communication is used to send commands to, and collect data from, underwater sensor nodes. Radio frequency communication is used to deliver the obtained data for a monitoring center.

Underwater sensor network architectures might be composed by heterogeneous nodes. It is important to mention that terrestrial wireless sensor networks have been using heterogeneous nodes to improve network performance [10], either by improving data delivery or by reducing energy consumption. In UWSNs, the use of heterogeneous nodes might also be used to improve the network performance or as a requirement of the application. In the former, for instance, UAVs can be used for better localization and time synchronization of ordinary underwater sensor nodes [11]. In the second, heterogeneous communication (e.g., acoustic and optic communication) can be used to improve the performance of video transmission from underwater nodes to the monitoring center [12].

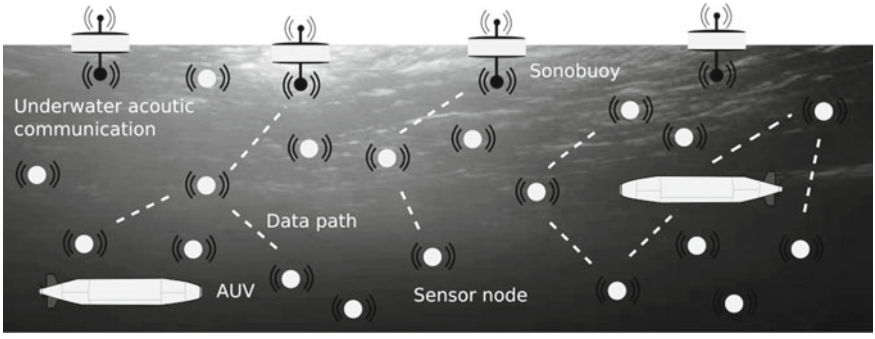


Fig. 1 An example of an underwater sensor network architecture

Frequently, a more specific architecture is deployed according to the characteristics of the considered application. In general, underwater monitoring applications might be categorized into two broad classes: *long-term monitoring* applications and *short-term monitoring* applications. In long-term monitoring applications, UWSNs are used to collect non-time-critical information about the area of interest and monitored events. This class of applications relies on event-driven or periodic data collection, which might result in very infrequent transmissions. Usually, a well-defined event at a particular spatial area is the target of the monitoring system. In terms of network architecture, the underwater nodes are attached to buoys or anchors to remain almost static in the desired location. Some aquatic monitoring applications of this category include: water quality, hydroelectric reservoir [13], marine biology [14], deep-sea archeology, seismic predictions [15], pollution detection [16], and oil/gas field monitoring [17].

In short-term underwater monitoring applications, UWSNs are frequently used to collect time-critical information in a 4D nature, that is spatial and temporal correlated data. By doing so, underwater sensor nodes are deployed without tethers. They freely drift according to the ocean currents. Because of the mobility and sparse deployments, the connectivity of the nodes will last for only a short period of time, which might demand for the deployment of additional nodes or the capture and redeployment of the previous ones. Some applications of this category include: underwater natural resource discovery, anti-submarine and target tracking mission [18], loss treasure discovery, tsunamis and seaquakes monitoring, oceanographic data collection [19], navigation assistance, and pollutant content monitoring [20].

3.2 High Cost

Nowadays, an underwater sensor network and its deployment are very expensive, being of the order of hundreds of thousands of dollars. As described in the following, two major factors contribute to high deployment and maintenance financial cost.

First, underwater sensor nodes and autonomous vehicles (AUVs) are naturally expensive, due to their acoustic modem transceiver and the appropriate hardware for protecting the circuitry. They have high energy consumption due to the characteristics of underwater acoustic communication, which can stop the underwater monitoring and exploration missions prematurely. Unlike WSNs, underwater sensor nodes must be collected at the end of the monitoring mission.

Second, ship missions are needed for the deployment and maintenance of the underwater nodes and network. Ship missions can last several days, since sensors must be attached to docks, anchored buoys, seafloors, autonomous underwater vehicles (AUVs), low-power gliders, or underpowered drifters, depending on the desired network architecture. Moreover, they are affected by external variables, such as weather conditions.

These two abovementioned factors drastically increase the financial cost of underwater network deployment and maintenance. Since underwater nodes have a limited

source of energy, i.e., they are operated by batteries, there is a need for the design of energy-efficient networking protocols to prolong the network lifetime and reduce the number of ship missions intended for network maintenance.

In addition, UWSNs are recognized by having a high energy consumption. The energy cost of transmitting a data packet is of the order of dozen of watts. Moreover, due to packet error and collisions, a generated packet may require some retransmissions until it reaches its destination. This high energy consumption is critical because it shortens the network lifetime. Thus, frequent ship missions should be employed for network redeployment and maintenance. Therefore, the design of energy-efficient routing protocols is required. Interested readers might find more information for reducing energy consumption of UWSNs in [21].

3.3 Involuntary Mobility

In some underwater sensor network applications, the sensor nodes will not be tethered to anchors or buoys. In fact, they will move according to the currents of the oceans. Mobility in the aqueous environment is very particular. It is affected by several environmental factors, such as water temperature, currents, boundary conditions, atmospheric forces, and bottom topography [22]. For instance, some underwater sensor nodes will move along a straight trajectory, while others will trap into flows and stay moving in loops [23], as observed from the analysis of the mobility traces of 46 floats drifting in the Mediterranean Sea.

As a matter of fact, the involuntary mobility of underwater sensor nodes will impair network connectivity and, consequently, the performance of networking protocols. This mobility pattern is so peculiar and depends on several factors that make its prediction challenging, as well as its use by networking protocols as usually happens in other scenarios of wireless ad hoc networks.

In this case, algorithms and protocols should be proposed for the reactive topology control in UWSNs. For instance, topology control, through power control, can be used to create long-range links to restore network connectivity. Moreover, topology control, through controlled mobility of some nodes to perform either depth adjustment of underwater nodes or controlled trajectories of UAVs, can better position some nodes aiming to restore connectivity between network partitions occasioned by involuntary mobility.

3.4 Underwater Acoustic Channel

The main aspect differentiating an underwater sensor network from terrestrial WSNs is the acoustic channel used for underwater wireless communication. Currently, the acoustic communication channel has been considered the most feasible technology

for underwater wireless communication. This is because the high-frequency radio waves are strongly absorbed underwater; also, optical waves suffer from heavy scattering and are restricted to short-range line-of-sight applications.

However, it is also recognized as one of the most challenging technologies for wireless communication, diminishing the performance of networking protocols in UWSNs. The underwater acoustic channel introduces novel challenges that are not experienced in terrestrial wireless networks. For instance, underwater wireless communications experience large and variable delays due to the low sound propagation speed of approximately 1500 m/s; temporary losses of connectivity due to the path loss; high noise that results in a high bit error rate; severely limited bandwidth due to the strong attenuation in the acoustic channel and multi-path fading; shadow zones; and the high communication energy cost, which is of the order of tens of watts.

As will be discussed in detail, underwater sensor network links are strongly impaired by high path loss, noise, multi-path signal propagation, limited bandwidth capacity, Doppler spreading, and high power consumption. In this environment, the frequency choice used for communication plays an important role. For instance, as can be seen in Fig. 2a, signal absorption is associated with frequency. Higher frequencies result in higher signal absorption. Conversely, the overall noise intensity affecting the communication decreases with an increase in frequency (refer to Fig. 2b). Moreover, noise intensity depends on specific frequency bands, as depicted in Fig. 2c. Another important characteristic of the underwater acoustic channel is that the bandwidth depends on the transmission power, radio frequency, and transmission distance. Figure 2d shows that the $1/A(l, f)N(f)$ factor apparently is a function of

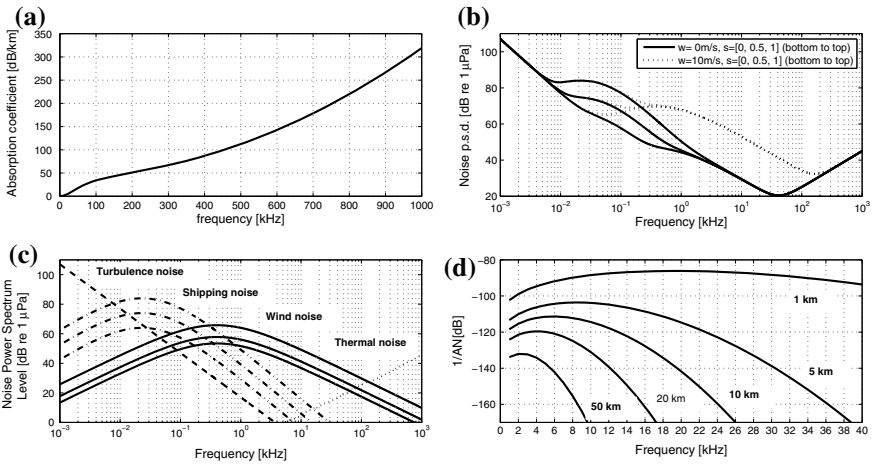


Fig. 2 Underwater acoustic channel characteristics: **a** Thorp's absorption coefficient. **b** Power spectral density of the ambient noise. **c** Power spectrum level based on empirical formula of different ambient noise source. **d** Signal-to-noise ratio depends on the frequency and distance through the factor $1/A(l, f)N(f)$

distance and frequency. For a more detailed overview about the characteristics of the underwater acoustic channel, the interested reader might refer to [24].

In a UWSN, the passive sonar equation is used to characterize the signal-to-noise ratio (SNR) of an emitted underwater signal at the receiver. Accordingly, the SNR is given as:

$$SNR = SL - A(d, f) - N(f) + DI \geq SINR_{\text{th}}, \tag{1}$$

where SL is the source level, $A(d, f)$ is the transmission path loss, $N(f)$ is the noise level, DI is the directivity index, and $SINR_{\text{th}}$ is a decoding threshold. In Eq. 1, all quantities are in dB re mu Pa.

The path loss $A(d, f)$ describes the attenuation of a single, unobstructed propagation path, over a distance d for a signal of frequency f, due to a large-scale fading. The path loss is mainly caused by the *geometrical spreading* and *signal attenuation* associated with frequency-dependent absorption. It is calculated as:

$$10 \log A(d, f)/A_0 = k \times 10 \log d + d \times 10 \log a(f), \tag{2}$$

where k is the spreading factor and $a(f)$ is the absorption coefficient. The absorption coefficient $a(f)$, in dB/km for f in kHz, is described by the Thorp's formula [25], given by:

$$10 \log a(f) = \frac{0.11 \times f^2}{1 + f^2} + \frac{44 \times f^2}{4100 + f^2} + 2.75 \times 10^{-4} f^2 + 0.003. \tag{3}$$

The geometry of propagation is described using the spreading factor k. Its commonly used values are $k = 2$ for spherical spreading, $k = 1$ for cylindrical spreading, and, for a practical scenario, k is given as 1.5.

In addition, a transmitted acoustic signal will be impaired by human-made and environmental noise. In general, we can observe four sources of noise [26]: turbulence (N_t), shipping (N_s), waves (N_w), and thermal noise (N_{th}). It is important to mention that each source will have greater impact based on a given frequency f. The overall ambient noise is $N(f) = N_t(f) + N_s(f) + N_w(f) + N_{\text{th}}(f)$. For a frequency f in kHz, shipping s ranging between 0 and 1 (light to dense) and wind in m/s, the four noise components in dB re mu Pa per Hz are given by:

$$\begin{aligned}
10 \log N_t(f) &= 17 - 30 \log f \\
10 \log N_s(f) &= 40 + 20(s - 0.5) + 26 \log f - 60 \log(f + 0.03) \\
10 \log N_w(f) &= 50 + 7.5 w^{1/2} + 20 \log f - 40 \log(f + 0.4) \\
10 \log N_{th}(f) &= -15 + 20 \log f.
\end{aligned} \tag{4}$$

Finally, it is important to mention that, conversely to what happens in WSNs, the bandwidth of the underwater acoustic channel depends on transmission power, radio frequency, and transmission distance. The $1/A(l, f)N(f)$ factor is a function of distance and frequency.

The aforementioned peculiar characteristics of the underwater acoustic channel make the use of networking protocols designed for terrestrial WSN impractical for UWSNs. In fact, some of the well-established principles, determined from decades of research in WSNs, are not even suitable for the design of networking protocols in UWSNs. For instance, data fusion is a good and indicated practice in WSN to reduce data communication and, consequently, energy consumption. To do so, routing paths are overlapped as much as possible at fusion nodes. These nodes are then responsible for aggregating several pieces of information, reducing the number of transmissions needed to deliver them to the sink node. Conversely, in UWSNs, the overlapping of routing paths toward a few central nodes will increase packet collision in their proximity, as well as quickly depleting their energy. These undesired effects will shorten the UWSN lifetime.

4 The Benefits of Topology Control in Underwater Sensor Networks

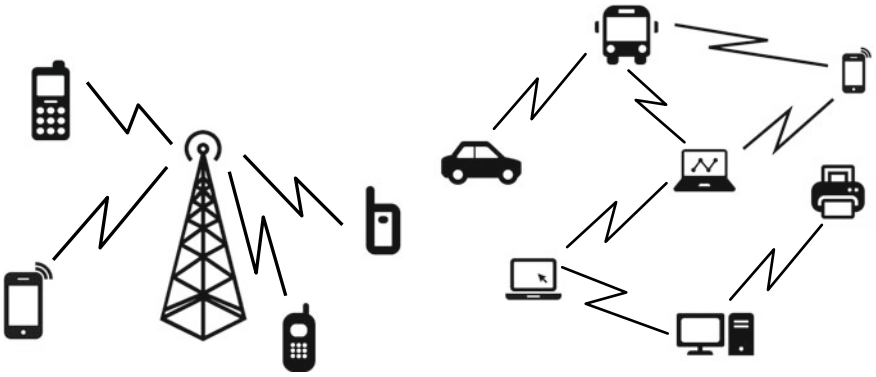
The topology of a network impacts on the performance of networking protocols and distributed services provided for the network. In summary, the network topology dictates how each node interconnects with each other. How the nodes are interconnected will affect the performance of networking protocols. For instance, routing protocols might have an increased performance when the interconnection between the nodes is done in such a way that we can easily have all of them interconnected to a central node (access point or base station) that can forward data packets between the nodes (e.g., infrastructured networks shown in Fig. 3a).

However, wireless ad hoc and sensor networks (refer to Fig. 3b) do not have such a central node for coordinating the networking tasks. For the majority of scenarios, we cannot even plan and control the nodes' deployment. More critically, the network topology in these applications might change because of uncontrollable events, such as hardware failures and breakdown.

More specifically, the topology in underwater sensor networks might change frequently due to the involuntary mobility of underwater nodes, described in Sect. 3.3, and time-varying link quality effects, described in Sect. 3.4. In fact, depending on network density, even a slow-moving coastal current will be enough to significantly change the network topology, diminish the connectivity, and make the sensor network unattainable after a few hours.

In this regard, topology control will be fundamental for improving the performance of underwater sensor networks. Topology control consists of the conscious and purposeful topology organization through the adjustment of some network parameters. It can be done proactively, to achieve a desired network property (e.g., connectivity), or reactively to stabilize and smooth the undesirable changes in the network topology, such as those due to the mobility.

In this chapter, we discuss topology control in UWSNs from the perspective of three broader categories: (i) power control, (ii) wireless interface management,



(a) Infrastructured network architecture (b) Ad hoc network architecture

Fig. 3 Architectural models of wireless networks

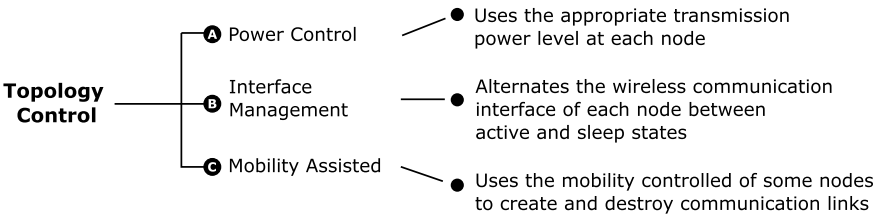


Fig. 4 Topology control in underwater sensor networks

and (iii) mobility-assisted-based topology control. This categorization is proposed based on the main technique used to control the network topology. The proposed categorization, as well as a short description of each approach, is summarized in Fig. 4 and Table 1.

4.1 Power Control-Based Topology Control

Transmission power control-based topology control is the most common approach used for the design of topology control algorithms. In this approach, topology control is performed either by choosing a unique transmission power level that will be used by all the nodes or through the assignment of the proper transmission power level for each node [27]. In the first approach, the minimal transmission power ensuring overall connectivity is selected. In the second approach, non-homogeneous transmission power level is assigned to each node. By using power control-based topology control, whenever a node increases or decreases its transmission power, the changes in the network topology happen through the creation/drop communication links by

Table 1 A summary of the techniques used to design topology control algorithms for UWSNs

Approach	Main idea	Advantages	Disadvantages
Power control	• The adequate transmission power is assigned to each node. Thus, communication links will be created and/or removed. Some algorithms aim to achieve a network-wide property, such as network connectivity	• Simple • Scalable • Conserves energy • Does not change the sensing coverage • Can overcome time-varying acoustic channel quality	• May diminish the network connectivity • Increases the number of hops and end-to-end delay
Wireless Interface mode management	• The wireless interface of nodes alternates between active, sleeping, and powered-off modes. This changes the interconnections between nodes over time	• Simple • Scalable • Conserves energy relative channel polling • Does not change sensing coverage	• Changes network density • Changes routing paths from time to time • Increases delay
Mobility-assisted	• Some mobile nodes are moved to new locations, creating new interconnections	• Improves network connectivity • Deals with network partitions • Improves data collection from hotspots	• Needs of trajectory planning procedures • Energy cost for mobility • May change sensing coverage

increasing and reducing their transmission power. This approach was extensively explored to reduce energy consumption in wireless sensor networks.

In general, power control-based topology control algorithms are proposed for reducing energy consumption. This is because short-range multi-hop communication spends less energy than long-range wireless communication, since the energy required for transmitting a data packet is proportional to the distance between the communicating pair of nodes [28]. The use of short-range communication links also reduces the energy consumption in UWSNs [29, 30]. However, in underwater acoustic sensor networks, this approach has mainly been considered to mitigate the drawbacks of underwater acoustic communication and improve network performance, that is, to increase link reliability and spatial reuse.

Power control-based topology control algorithms in UWSNs increase the network throughput by improving either transmission scheduling algorithms or the achievable bandwidth. In the former case, power control assigns an adequate communication range to the nodes in order to increase spatial reuse and, consequently, network throughput. In the latter case, power control determines the appropriate communication range and corresponding optimal frequency to increase the achievable bandwidth.

It is well known that the throughput/goodput of network systems is closely related to the spatial reuse, which in turn represents the number of links that can simultaneously transmit data [31]. In wireless networks, a critical factor limiting the network capacity is the interference between spatially closer nodes. Due to the broadcast nature of the communication medium in these networks, nodes located inside an interference zone of both the transmitter and receiver should be kept silent (i.e., without transmitting) when a transmission is happening. This silence is necessary to avoid message collisions and, consequently, waste of network resources. In UWSNs, the collision-free transmission coordination among nodes is challenging. As a matter of fact, the design of efficient medium access control (MAC) protocols for UWSNs is daunting, mainly due to high message propagation delay occasioned by the low acoustic signal propagation speed. Therefore, the effectiveness of using feedback solutions for MAC protocols (e.g., carrier sensing and RTS/CTS exchange) is not common in UWSNs [32]. In fact, contention-free-based approaches (e.g., FDMA, CDMA, or TDMA) could lead to a better performance of UWSNs [33].

In this context, power control-based topology control can confine interference and improve spatial reuse, by properly assigning a short-range communication for the nodes. The use of fixed long-range communication links is disadvantageous, as it does not optimize the total number of concurrent transmissions of the network. In this scenario, topology control through transmission power control can maximize the spatial reuse, as it is helpful in confining interference while ensuring network connectivity and link reliability.

Finally, the design of topology control algorithms based on power control can be used to explore the bandwidth–distance relationship of the underwater acoustic channel for improving the performance of UWSNs. In the underwater acoustic channel, the distance between the communication nodes will also affect the useful bandwidth. In fact, for a given communication distance, there is a constraint on the maximum usable frequency. This constraint is due to the signal absorption, which increases rapidly as the frequency increases. Moreover, the intensity of the ambient noise (e.g., turbulence, shipping, waves, and thermal noise) impacting the underwater acoustic signal varies according to the used frequency. Therefore, there is also a limitation of the useful acoustic bandwidth from below, as the noise is high for low frequencies and decays as soon as the frequency increases. This leads to the conclusion that, based on the attenuation-frequency product, there is an optimal frequency f_c for each communication distance in which impairments on the transmitted signal are minimized [26].

4.2 Wireless Interface Management-Based Topology Control

Another classical approach for topology control is through density control, by means of the wireless interface management. It is important to mention that in wireless

network systems, the communication interface of the nodes has some operational states: transmitting, receiving, idle, sleep, and, of course, totally powered-off. Hence, each of these operational modes has different unitary costs of energy. Therefore, the overall energy consumption of a node is determined from the amount of time it spends with its wireless interface in each mode, as given by the equation below:

$$E_{consumption} = T_{tx}e_{tx} + T_{rx}e_{rx} + T_i e_i, \tag{5}$$

where T_{tx}, T_{rx}, and T_i are the amount of time that the wireless interface spent transmitting, receiving, and in the idle state, respectively, and e_{tx}, e_{rx}, and e_i are the energy consumption of transmitting, receiving, and idle states per time, respectively.

The wireless interface management topology control approach is highly suitable for wireless sensor networks in general. Usually, data transmission in sensor networks happens infrequently (e.g., once a day or even a week in UWSNs [34]). Therefore, it is not necessary to have all the nodes awake all the time.

Topology control algorithms using this approach are mostly designed to conserve energy. The idea is to have the nodes in the powered-off operational mode as much as possible. Thus, it can conserve energy during times of no communication. By doing so, this methodology of topology control changes the network topology by either keeping a backbone of active nodes for communication purposes, where the remaining redundant nodes can go to sleep (density control), or by controlling the active period of the nodes to make them available at the same time, enabling communication (duty-cycling operation). Both approaches lead to energy conservation, since they reduce the amount of time that a node unnecessarily spends listening to the channel.

Moreover, by means of density control, this topology control approach might reduce communication interference, which decreases message collisions and the need for message retransmissions. Consequently, wireless interface management-based topology control will also improve the performance of medium access control protocols in UWSNs and decrease the overhead of neighboring discovery of routing protocols, since only a subset of the nodes will be awake.

Using this methodology, topology control can be performed either through the network density control or through duty cycling, as described in the following. In density control-based topology control, a subset of the deployed nodes is selected to be in active mode and is used as a communication backbone for multi-hop data delivery. In order to conserve energy, the unchosen nodes can change their wireless interface state for the sleep mode. The density control-based topology control approach has been frequently employed in terrestrial wireless sensor networks (WSNs), since they usually comprise high-density deployments. Nowadays, the density control-based topology control approach has been less explored in underwater sensor networks. The main reason for this is the fact that underwater sensor networks typically involve sparse deployments. Thus, in UWSNs, each node has a vital role for network connectivity.

However, as will be pointed out in the following, there are some proposals that investigate the benefits and drawbacks of this topology control methodology in UWSNs. Further investigations might be motivated by the facts that (i) the

deployment of underwater sensor nodes usually occurs according to the following manner: a group of underwater drift nodes are initially deployed in a small area, and they spread out with ocean currents as the time goes by, which creates a high-density deployment at the beginning of the monitoring mission; and (ii) underwater nodes might use a high communication range, which will result in redundancy.

The second wireless interface management-based topology control approach consists in having the nodes operate in a duty-cycled manner. Accordingly, each node periodically alternates the state of its wireless interface between active and sleep modes. In this approach, energy conservation is achieved by avoiding keeping the nodes idly listening to the channel, especially in low traffic load scenarios of sensor networks. In general, duty-cycling protocols have been classified as synchronous and asynchronous, according to the mechanism used to ensure the pairwise communicating nodes awake at the same time, i.e., during the message transmission. In synchronous duty cycling, the sleep/wake-up schedule of the nodes is aligned, by means of an explicit negotiation. Therefore, a sender node is always aware of the time that its neighbors will be awake and apt to receive data messages. In asynchronous duty cycling, conversely, the duty cycle schedule of the nodes is completely decoupled. Therefore, at an arbitrary moment in time, only a subset of nodes might be in the active mode, i.e., awake and apt to receive data messages.

Synchronous and asynchronous duty-cycling protocols have peculiar characteristics that will impact the UWSN performance. Synchronous duty-cycling protocols entail periodic signaling to ensure an alignment between the duty-cycling schedules of the nodes. Moreover, there is the need for synchronized clocks, which require extra computing and communication efforts. This approach seems to be unfeasible in UWSNs because of the overhead for clock synchronization and schedule exchanges. Asynchronous duty-cycling protocols, on the other hand, are simple and do not involve periodic beaconing for schedule alignments. However, these protocols lead to an increasing end-to-end delay, since there is no guarantee that a receiver node will be awake when a sender node has a data message to transmit. This is the case even though protocols falling into this category seem to be more suitable for the harsh environment of UWSN applications.

It is worth mentioning that duty cycling has been well investigated in wireless sensor networks (WSNs), but obtained insights and principles design cannot be directly applied in UWSNs. Traditional approaches for the design of asynchronous duty-cycling protocols in UWSNs will not perform well. The use of preamble transmissions before a data packet transmission, in order to guarantee that the sender and receiver will awake at the same time for the data transmission, is impractical in UWSNs because of the high energy cost for transmissions (of the order of dozen of watts). Moreover, recent approaches of joint design of opportunistic routing and duty-cycling protocols in WSN, in order to reduce energy cost and delay, cannot be applied in UWSN because of the poor link quality of the acoustic channel.

In this regard, the authors in [35] proposed and investigated the joint design of duty-cycling-based topology control and opportunistic routing in UWSNs. Their proposed model allows us to study the performance of the three common methodologies: low-power listening, low-power probing, and naive asynchronous duty cycling, used

for the design of asynchronous duty-cycling protocols, in the scenario of opportunistic routing in underwater networks. As expected, it was possible to note how duty cycling impairs the network connectivity in already sparse UWSN deployments.

In addition, the authors in [36] investigated how the sleep interval at each node could be used to achieve a balanced energy consumption. Balanced energy consumption is important in order to guarantee that all the nodes will deplete their battery approximately at the same time. When the batteries of central nodes in UWSNs are quickly depleted [37], the UWSN will experience disconnections and, consequently, poor performance for the application, since some of the collected data will not be delivered to the monitoring center due to the lack of routing paths.

4.3 Mobility-Assisted-Based Topology Control

A recent concept being considered for topology control in underwater sensor networks consists of the exploration of controlled mobility of certain underwater nodes. In the mobility-assisted topology control approach, the basic idea is to move one or a few underwater nodes for new locations.

In the terrestrial wireless sensor network, mobility-assisted-based topology control has been extensively designed for network connectivity restoration [38]. This is motivated by the fact that, in these networks, the network connectivity is frequently disrupted by node failures occasioned by either battery depletion or problems in the hardware. In these scenarios, the topology control algorithm performs two basic steps. First, new locations are computed for some mobile nodes. Second, these nodes are moved to the determined locations and act as bridges between the two previously disconnected network segments.

In the context of UWSNs, mobility-assisted-based topology control algorithms have been used to improve data collection [39], data routing [40–45], coverage [46], or even underwater node localization [47]. In these networks, current nodes with depth adjustment capabilities, as well as underwater autonomous vehicles (UAVs), could be used for organization of the topology. Ordinary underwater sensor nodes with depth adjustment capabilities are more affordable than the use of UAVs. However, the topology control might be limited, given that these nodes can only move vertically. In contrast, UAVs allow for efficient topology control, as they move in any desired direction. However, the use of a few units might be impractical for several applications, because of their high monetary cost.

The mobility-assisted-based topology control is probably the most powerful methodology for changing the network topology. We can state that the previously discussed methodologies make “soft” changes in the network topology by enabling or failing to enable some communication links. In fact, there are no physical changes regarding the location of the nodes, limiting the possible neighboring nodes that a given node can have. Conversely, using mobility-assisted-based topology control, nodes can be relocated and, as a result, a more effective density control can be accomplished and new sets of neighboring nodes can be enabled. Therefore, this approach

can achieve better performance when topology control is used to improve coverage, among other applications.

Topology control using a mobility-assisted-based approach must carefully select the nodes to be moved and the location to which they will be moved. In most network deployments, there is an associated cost in moving mobile nodes, such as the energy cost. These costs should be considered when the topology control algorithm selects the nodes and new locations for them. This is to prevent mobile nodes from quickly depleting their batteries, which might incur disruption in the network connectivity. Therefore, in UWSNs, independent of the use of UAVs or underwater nodes with depth adjustment capabilities, the main research challenge of designing mobility-assisted-based topology control is to design efficient protocols for selecting which nodes are to be moved and determining their new locations.

5 Geographic and Opportunistic Routing in Underwater Sensor Networks

Data routing is another critical challenge in UWSNs. This is due to the peculiar characteristics of the underwater acoustic channel. Moreover, data routing is severely impaired by network topology. Designing routing protocols for UWSNs is challenging, due to the spatiotemporal-varying nature of the acoustic link quality that unpredictably changes the network topology.

In this scenario, geographic and opportunistic (geo-opportunistic) routing have been shown to be preferable for UWSNs. Geographic routing does not require the establishment and maintenance of end-to-end routing paths from each sender node to the destinations, as may occur in proactive and reactive routing paradigms. In this approach, a forwarder node, aware of its geographic location and the location of its neighboring nodes, transmits a message to a locally optimal next-hop node closest to the destination (greedy forwarding strategy). Therefore, there is no need for excessive and systematic flooding for route discovery and maintenance, which would diminish the network performance due to the overhead and message collisions, and result in a large waste of energy.

A key factor preventing geographic routing from being a better solution for UWSNs is the inability of the protocol to handle acoustic links with low reliability. In this regard, opportunistic routing (OR) is proven to be a promising paradigm for the design of routing protocols for UWSNs. In traditional multi-hop routing, a packet is transmitted to a specific next-hop node using a unicast communication. If the next-hop node does not receive it, the packet should be retransmitted. After a finite number of unsuccessful retransmissions, the packet is discarded. In OR, a set of candidate nodes is involved for advancing the packet toward the destination. Accordingly, a packet is sent leveraging the broadcast nature of the wireless transmission. The candidates that receive the packet will continue, coordinately, forwarding it in a prioritized way, such that a low-priority node transmits the packet if none of the

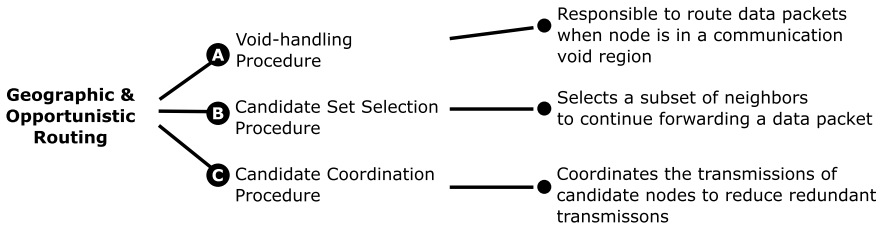


Fig. 5 Building blocks of routing protocols for underwater networks

high-priority nodes have done so. Therefore, a packet is retransmitted only if none of the candidates have received it.

Opportunistic routing paradigm has benefits and drawbacks that must be considered when it is used to design routing protocols for underwater networks. OR increases packet delivery and decreases the number of packet collisions, since the probability that at least one candidate would correctly receive the packet is high compared to the traditional unicast routing. However, the packet delivery end-to-end delay is high due to the nodes transmission coordination. The harsh underwater communication environment can result in poor transmission nodes coordination, culminating in redundant packet transmissions, increasing packet collisions, and delay and energy consumption. Moreover, the assignment of the same high priority for some candidates may deplete their energy sooner, leading to partitions in the network.

Generally, the design of geo-opportunistic routing protocols for UWSNs concerns three main building blocks [48], as shown in Fig. 5. First, it should present a recovery procedure to route data packets when the greedy forwarding strategy fails to do so. Second, it must have a candidate set selection procedure that will select the most suitable neighboring nodes to participate in the task of data forwarding toward the destination. Third, geo-opportunistic routing protocols should implement a candidate coordination procedure. This procedure is responsible for the data transmission coordination of the candidates, to avoid redundant packet transmissions. In the following, we discuss each of these procedures in detail. Table 2 shows an overview of some underwater sensor network routing protocols and their properties. At this moment, there is a lack of works presenting quantitative comparison of recent routing protocols for underwater sensor networks. It is worth noting that only a few of them are opportunistic protocols. However, we include them because it is still necessary to investigate their next-hop forwarder node selection mechanism. Moreover, they can be adapted to the OR paradigm.

5.1 Void-Handling Procedure

Geographic routing entails one-hop neighboring information to forward data messages. The neighboring node closest to the destination is selected to continue

Table 2 Routing protocols for underwater sensor networks

Routing protocol	Candidate set selection	Candidate coordination
VBF [49]	Receiver-side-based	Timer-based
HH-VBF [50]	Receiver-side-based	Timer-based
DFR [51]	Receiver-side-based	
DBR [52]	Receiver-side-based	Timer-based
VBVA [53]	Receiver-side-based	Timer-based
HydroCast [54]	Sender-side-based	Timer-based
DCR [40]	Sender-side-based	
VAPR [55]	Sender-side-based	Timer-based
GEDAR [42]	Sender-side-based	Timer-based
SBR-DLP [56]	Hybrid-based	
CARP [57]	Hybrid-based	
Nowsheen et al. [58]	Hybrid-based	
LASR [59]	Sender-side-based	
CDRP [60]	Sender-side-based and hybrid-based	Timer-based
DARP [61]	Receiver-side-based	Timer-based
REBAR [62]	Receiver-side-based	
EEDBR [63]	Sender-side-based	Timer-based
QELAR [64]	Sender-side-based	
FBR [65]	Hybrid-based	

forwarding the data packet. Therefore, this routing approach is impaired by the unpredictable and dynamic changes in the UWSN topology. For instance, based on the location and interconnection of the nodes, communication void regions may appear in the network [66, 67].

Figure 6 shows an example of the communication void region problem. Communication voids happen whenever a sender node (void node) is located at a maximum local; i.e., it does not have a neighboring node in closer proximity to the destination. When a data message reaches such void nodes, it should be routed through an alternative path, or discarded. Often, void-handling procedures have been employed by geographic routing protocols in UWSNs. A void-handling procedure is used for routing data messages from void nodes to a node that can resume the greedy forwarding strategy. Since data messages are routed from void nodes, instead of being discarded, void-handling procedures avoid the degradation of the network performance and application.

In the literature, there are three main methodologies used to design void-handling procedures for geo-opportunistic routing in UWSNs [68]: (i) bypassing void region approach, where data packets are routed along alternative routing paths circumventing the void region, (ii) power control approach, where void nodes increase their

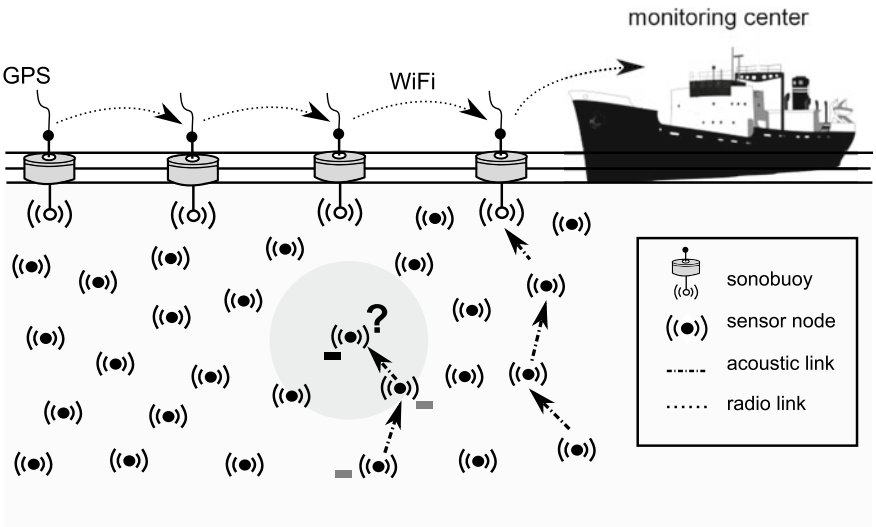


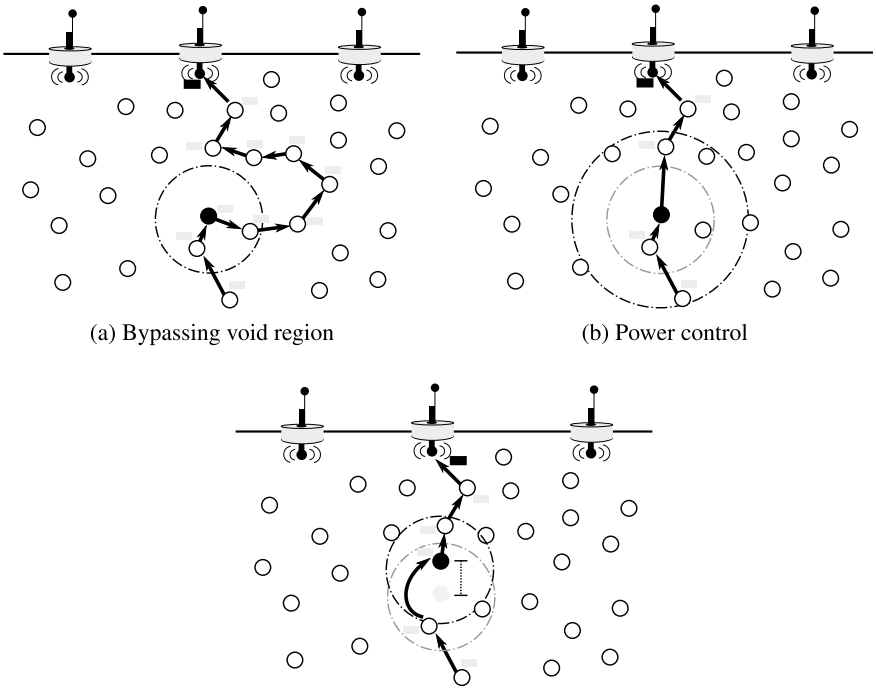
Fig. 6 Communication void region problem

communication range, and (iii) depth adjustment approach, where void nodes are moved to new locations.

Using the bypassing void region methodology (Fig. 7a), void-handling procedures determine an alternative path to route data packets. The idea is to find and maintain an alternative path that circumvents the void area. The main advantage of this approach is that it does not change the covered area of the network; that is, it does not modify the location of the nodes. However, it might entail long routing paths that increase the number of times that a packet is transmitted. Therefore, there will be high costs associated with delivering packets to void nodes and greater probability of packet error and collisions.

Using the power control methodology (Fig. 7b), void nodes may increase its transmission power to find a neighboring node capable of continuing to forward the data packet to the destination. This approach is simple and scalable. It also does not change the initial location of the nodes. The use of long-range communication links, however, will increase the energy consumption and result in more packet collisions, as discussed in Sect. 4.1.

Using the depth adjustment methodology (Fig. 7c), void nodes are moved to new locations. Herein, the idea is to move void nodes for new depths in order to find a new set of neighboring nodes where one of them can continue forwarding data packets by means of the greedy forwarding strategy. A more generalized approach may use underwater autonomous vehicles to visit void nodes and collect data packets from them. The main disadvantage of this approach is that the initial deployment of the nodes changes as some of them are moved to new locations. Moreover, all nodes may be closer to the surface if their depth adjustment is not performed in an



(a) Bypassing void region

(b) Power control

(c) Depth adjustment

Fig. 7 Methodologies for the design of void-handling procedures

efficient manner. However, the energy cost is less than the other two approaches when inflate buoys are used to control the vertical movement of the nodes (e.g., Drogue nodes [69]).

In summary, each methodology has advantages and disadvantages that should be considered when designing void-handling algorithms for UWSNs. The network density and application requirements may drive the design of void-handling algorithms. For instance, if only a few fractions of the nodes are in a void region, we could consider power control or bypassing void region. However, in sparse deployments, depth adjustment might prove a better strategy to avoid increasing the energy at central nodes.

5.2 Candidate Set Selection Procedure

Candidate set selection is the first procedure to be used when a node has a data packet to send. This procedure is responsible for selecting a subset of the neighboring nodes to continue forwarding the packet toward the destination. Usually, a fitness function (e.g., the expected packet advancement), is employed to determine which neighbors

Table 3 A summary of candidate set selection methodologies

Technique	Approach	Advantages	Disadvantages
Sender-side	Next-hop forwarder candidate nodes are determined in the current transmitter, from the neighborhood information and state of neighboring nodes	• Possibility of complex fitness function for next-hop selection • Better prioritization of candidate nodes	• Beaconing for neighborhood discovery
Receiver-side	Each receiver node is responsible to determine if it is a next-hop candidate node from a given information in the packet header	• Simple and scalable • No overhead for neighborhood discovery	• Poor prioritization and transmission coordination of the next-hop candidates • High redundant packet transmissions
Hybrid	Transmitter node requests one-hop neighborhood information but only neighbors meeting a given criterion (e.g., a certain battery level) will replay as a potential candidate node	• Neighborhood is discovered on demand • Suitable for low traffic load UWSN applications	• Increases the already high end-to-end delay

are suitable to be a next-hop forwarder. Moreover, the fitness function is also considered to prioritize next-hop candidate nodes. This prioritization is fundamental to determine the order in which each candidate node will forward the received data packet.

During the candidate set selection procedure, the considered neighbors are sorted according to their suitability (fitness value), and their unique addresses are then included in the next-hop candidate set. Herein, it is important to mention that only a few of candidate nodes are needed to participate in the data forwarding processes. Actually, the inclusion of a large number of candidates does not improve the packet delivery ratio [70]. Therefore, the inclusion of candidate nodes in the set eventually becomes subject to restriction, such as a limited number of candidates or a one-hop candidate set delay.

In [48], the authors proposed to the classification of candidate set selection procedures in sender-side-based, receiver-side-based, or hybrid-based procedures. In the following, we present detailed information about each category. Table 3 summarizes the candidate set selection methodologies.

5.2.1 Sender-Side-Based Candidate Set Selection

In sender-side candidate set selection procedures, the current forwarding node is responsible for selecting the next-hop forwarder candidate set. It selects the next-hop candidate nodes based on neighborhood information. Usually, neighboring information is acquired through periodic beaconing. Based on the neighborhood information and on the next-hop selection metric or fitness function, the current forwarder node determines which neighbors are able to continue forwarding the data packet toward the destination.

Then, the next step is to rank the capable nodes and include them into the next-hop candidate set, according to their priorities. Finally, the unique ID of the chosen nodes is included in the packet header to inform them that they were selected as next-hop forwarder nodes. Frequently, a bloom filter or membership checking data structures are used to avoid long packet headers, which would increase the packet error rate.

In geo-opportunistic routing protocols using sender-side-based candidate set selection, as the neighborhood is known in advance, more complex and multi-objective fitness functions can be used to select the next-hop candidate set. Therefore, application and underwater acoustic channel characteristics can be considered for the candidate set selection. For instance, buffer and distance information of the neighbors can be used by the next-hop candidate set selection procedure in a real-time application (e.g., oil spill monitoring) to ensure an acceptable packet reception ratio with a limited delay.

Moreover, in order to mitigate the problem of hidden terminals, only nodes close enough to hear one another's transmission can be selected to belong to the next-hop candidate set [54]. The main drawback of this approach stems from the need to keep updated neighborhood information at the nodes, considering the node mobility caused by ocean currents. The use of beacon messages can overload the channel and increase packet collisions, because of the slow propagation through acoustic channels.

5.2.2 Receiver-Side-Based Candidate Set Selection

In receiver-side candidate set selection procedures, each neighboring node is responsible for determining whether it is a next-hop candidate node for the received data packet. To do so, the current forwarder only includes a piece of information in the packet header (e.g., its depth or distance to the destination) and then broadcasts it.

From the information contained in the packet, each receiver node then determines locally whether it is a next-hop candidate node. This verification happens according to the rule adopted by the protocol (e.g., depth information or distance to the destination comparison). If the neighboring node is a candidate, it locally determines its forwarding priority and forwards the packet if and when it should do so.

Receiver-side-based next-hop candidate set selection procedure is simple and scalable. It does not require any neighborhood information. Therefore, it may contribute to energy conservation and increasing underwater acoustic channel utilization, as

there is no need for periodic beaconing or any information exchange during the data routing process.

Moreover, receiver-side-based candidate set selection is indicated for high traffic load applications (e.g., pollution monitoring), since it will not entail competition between control packets for access to the acoustic channel and colliding with the data packets. However, the candidate coordination and redundant packet suppression can be inefficient, leading to a high number of duplicated packet transmissions, such as in the DBR protocol [52], which consumes unnecessarily energy and does not provide any innovative information to the destination, being discarded when it is reached.

5.2.3 Hybrid-Based Candidate Set Selection

In hybrid candidate set selection procedures, the next-hop forwarder candidate set is determined by the current forwarder node and its neighbors, in two distributed steps. First, when a node has a packet to transmit, it informs its situation and requests information (e.g., battery level or link reliability) from its neighborhood through a broadcast of a control packet. Neighboring nodes that meet the established criteria (e.g., positive progress to the destination) will respond to the current forwarder node with the requested information. Second, the current forwarder node selects the next-hop candidate set based on received packets.

In the solutions belonging to this category, the neighborhood condition is known on demand, in contrast to the procedures based on the sender-side-based candidate set selection, where beaconing happens periodically. Moreover, short packets to the next-hop candidate selection and coordination are less likely to be lost, resulting in an improvement in candidate coordination performance.

This approach is highly suitable for low traffic load monitoring applications, such as periodic ocean temperature or salinity monitoring applications. However, this two-way procedure can increase the already high end-to-end end delay, due to the slow propagation through underwater acoustic channels.

5.3 Candidate Coordination Procedure

In opportunistic routing, the coordination of the transmission of the next-hop forwarder candidate nodes is a crucial component and may severely impact on the performance of an underwater sensor network. In general, with the inclusion of a suitable number of candidate nodes in the candidate set, one-hop link reliability improves and, therefore, the number of transmissions required to deliver a packet is reduced. However, in order to avoid redundant packet transmissions, candidate nodes must forward the packet in a coordinated manner. Basically, a low-priority node will transmit a received data packet only if high-priority nodes fail to do so. In this way, the number of transmissions of unnecessary and redundant packets is

Table 4 A summary of candidates' transmission coordination methodologies

Technique	Approach	Advantages	Disadvantages
Timer-based	A packet holding time is calculated at each candidate node. The defer time is proportional to the priority level of the candidate	• Simple and does not incur in network overhead • Better prioritization of candidate nodes	• Increased end-to-end delay
Control packet-based	Acknowledgment packets are used to coordinate the transmission between candidate nodes	• Can achieve better coordination and reduce redundant packet transmissions	• Network overhead

reduced, which will consume energy and fail to provide any additional information, being discarded at the destination.

It is important to mention that the coordination of candidate transmissions is a more challenging problem in underwater networks than in terrestrial wireless sensor networks. In the latter, a candidate set with only two or three nodes may be sufficient for obtaining a high packet delivery probability. Conversely, in the former, this number should be higher due to high noise, signal attenuation, and shadow zones.

Basically, two approaches have been considered for the candidate transmission coordination procedure in UWSNS: timer-based prioritization and packet control transmission for candidate coordination. In the following, we discuss each one of these two approaches. Table 4 summarizes the candidates' transmission coordination methodologies.

5.3.1 Timer-Based Candidates' Transmission Coordination

Upon receiving a packet in the timer-based coordination procedure, each candidate node holds it for a period of time or a number of transmission slots, according to its priority. The candidate will suppress its transmission if it receives an indication during its waiting period that the packet was forwarded by a high-priority node. Usually, this indication is the reception of the same packet, now coming from a high-priority node or an acknowledgment (ACK) packet. Otherwise, the node forwards the packet when its holding time expires.

The main advantage of this approach is the absence of extra control packets, which can further degrade network performance. However, as no control packet is used to coordinate and inform low-priority nodes, duplicated data packet transmissions can occur when the nodes are far from one another and cannot hear transmissions. In this case, duplicated data packet transmissions have a more negative impact on network

performance and energy consumption than the use of control packets, due to their differences in size.

5.3.2 Control Packet-Based Candidates' Transmission Coordination

Using control packet-based candidate coordination procedure, a candidate node upon receiving a packet will respond with a short control packet if high-priority candidates have failed to do so. The control packet transmission notifies the current forwarder node of the successful receipt of the packet and informs the other low candidate nodes that they should suppress their transmissions. Usually, control packet-based candidate coordination approaches designed for terrestrial wireless networks fall into one of the categories below: (i) *ACK-based*, in which the current forwarder node sends the data packet and the candidates respond with an acknowledgment packet according to their priorities and (ii) *RTS/CTS-based*, in which the current forwarder node sends an RTS packet, the candidates respond with a CTS packet according to priority, and finally, the data packet is sent to the candidate that replied to the RTS transmission.

The design principles mentioned above have severe drawbacks when applied to underwater sensor network scenarios. ACK-based approaches require all candidates close enough to the transmission to be heard by others. Otherwise, duplicated packet transmissions will take place and degrade network performance. RTS/CTS-based candidate coordination might perform poorly in underwater sensor network scenarios, since a high-priority node may successfully receive a short RTS packet, become the next-hop forwarder responding with the CTS packet, and eventually become unable to receive the data packet, as it is longer than RTS and its delivery probability is significantly lower.

6 Future Research Directions

In this chapter, we have discussed the benefits and challenges of topology control and opportunistic routing to improve the performance of large-scale deployments of underwater sensor networks. In fact, the design of these protocols has been an active area of research. Several advancements have recently been achieved in which various protocols were developed for non-mobile and mobile underwater sensor network scenarios. However, several directions of research with open problems still require further exploration. In the following, we point out three important and challenging future research directions.

First, there is the need for analytical frameworks for performance evaluation of topology control and opportunistic routing in underwater sensor networks. Analytical models are fundamental to obtaining insights that will further guide the design of networking protocols and distributed algorithms. Hence, such modeling research is

required in the context of UWSNs, since the principle designs employed in terrestrial WSNs cannot be directly applied in UWSNs.

Through mathematical models of opportunistic routing, for instance, we will be able to identify a suitable number of candidate nodes ensuring high delivery rates, as well as the best communication void region handling strategy for the considered scenario. In term of topology control, for instance, mathematical models are useful for determining a suitable sleep interval value in duty-cycled UWSNs, taking into consideration the conflicting goals of energy savings and network connectivity.

Second, there is a need for the design of topology control and opportunistic routing protocols for envisioned scenarios of Internet of Underwater Things (IoUT). Currently, the design of such protocols considers homogeneous scenarios of underwater sensor networks. In a few cases, UAVs are considered in conjunction with ordinary underwater sensor nodes. However, IoUT scenarios will be characterized by the existence of underwater devices with heterogeneous capabilities. Herein, IoUT devices will not only be heterogeneous in terms of processing, storage, or mobility, but they will also be heterogeneous in terms of communication. For instance, some of the nodes may be equipped with optical communication, while others will be communicating through the acoustic channel. Therefore, future design of topology control and opportunistic routing protocols should consider this heterogeneity to avoid overwhelming critical nodes, which will lead to network partitions and, consequently, short network lifetime.

Third, future research should look at efficient ways to perform topology control without excessive overhead. Due to the acoustic channel, overhead significantly diminishes the UWSN performance, as the overhead results in packet collision and increased energy consumption. A possible future direction for avoiding excessive overhead is the design of localized topology control solutions. In this approach, instead of a topology control considering the entire topology, algorithms may perform the topology organization only in the vicinity of critical nodes. For instance, to improve data routing, topology control may be performed only in the proximity of void nodes. In this way, control packets transmitted to acquire neighboring information will be confined to a few parts of the network.

7 Concluding Remarks

In conclusion, underwater sensor networks (UWSNs) are expected to revolutionize our methods for data collection in ocean environments. These networks will be fundamental for human beings to develop a better understanding of aquatic environments, for the proper exploration and protection of them. In this chapter, we discussed the main characteristics differentiating underwater sensor networks from the well-investigated terrestrial wireless sensor networks. In doing so, we highlighted peculiar characteristics of UWSNs that make it impractical to use the well-established principles on wireless sensor networks for the design of networking protocols. Hence, we provided a detailed discussion about advancements in topology control and

opportunistic routing to improve the performance of UWSNs. Finally, we pointed out some future research directions that must be investigated to achieve high deployments of large-scale UWSNs.

References

1. Woods Hole Oceanographic Institution. Know your ocean. http://www.whoi.edu/know-your-ocean/ (2017)
2. Rossby, T., Dorson, D., Fontaine, J.: The RAFOS system. J. Atmos. Ocean. Technol. **3**, 672–680 (1986)
3. Ocean Networks Canada. Ocean observing systems. http://www.oceannetworks.ca/innovation-centre/smart-ocean-systems/ocean-observing-systems (2017)
4. NOAA—Office of Response and Restoration. Deepwater horizon oil spill. http://response.restoration.noaa.gov/deepwater-horizon-oil-spill (2017)
5. Boukerche, A.: Algorithms and Protocols for Wireless Sensor Networks, vol. 62. Wiley-IEEE Press (2008)
6. Boukerche, A.: Algorithms and Protocols for Wireless, Mobile Ad Hoc Networks, vol. 77. Wiley-IEEE Press (2008)
7. Akyildiz, I.F., Pompili, D., Melodia, T.: Underwater acoustic sensor networks: research challenges. Ad Hoc Netw. **3**(3), 257–279 (2005)
8. Argo. About argo. http://www.argo.ucsd.edu/ (2017)
9. National Oceanic and Atmospheric Administration. How much of the ocean have we explored? http://oceanservice.noaa.gov/facts/exploration.html (2017)
10. Guidoni, D.L., Mini, R.A.F., Loureiro, A.A.F.: On the design of resilient heterogeneous wireless sensor networks based on small world concepts. Comput. Netw. **54**(8), 1266–1281 (2010)
11. Vieira, L.F.M., Lee, U., Gerla, M.: Phero-trail: a bio-inspired location service for mobile underwater sensor networks. IEEE J. Sel. Areas Commun. **28**(4), 553–563 (2010)
12. Han, S., Chen, R., Noh, Y., Gerla, M:. Real-time video streaming from mobile underwater sensors. In: Proceedings of the International Conference on Underwater Networks & Systems (WUWNET), pp. 21:1–21:8 (2014)
13. Vieira, L.F.M., Vieira, M.A.M., Pinto, D., Nacif, J.A.M., Viana, S.S., Vieira, A.B.: Hydronode: an underwater sensor node prototype for monitoring hydroelectric reservoirs. In: Proceedings of the 7th ACM International Conference on Underwater Networks and Systems (WUWNet), pp. 43:1–43:2 (2012)
14. Risch, D., Parks, S.E.: Biodiversity assessment and environmental monitoring in freshwater and marine biomes using ecoacoustics. Ecoacoustics: The Ecol. Role Sounds, p. 145 (2017)
15. Dotsenko, S.F., Eremeev, V.N.: Analysis of the necessity and possibility of tsunami early warning on the black-sea coast. Phys. Oceanogr. **18**(5), 288–296 (2008)
16. Khan, A., Jenkins, L.: Undersea wireless sensor network for ocean pollution prevention. In: Proceedings of the 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE), pp. 2–8, Jan 2008
17. Shukla, A., Karki, H.: Application of robotics in offshore oil and gas industry–a review part II. Robot. Auton. Syst. **75**, 508–524 (2015)
18. Yu, C.H., Min, S.H., Choi, J.W.: Sensor localization-based distributed target tracking filter in underwater sensor networks. In: Proceedings of the 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), pp. 706–711, July 2015
19. Vasilescu, I., Kotay, K., Rus, D., Dunbabin, M., Corke, P.: Data collection, storage, and retrieval with an underwater sensor network. In: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys), pp. 154–165 (2005)

20. Kato, N., Choyekh, M., Dewantara, R., Senga, H., Chiba, H., Kobayashi, E., Yoshie, M., Tanaka, T., Short, T.: An autonomous underwater robot for tracking and monitoring of subsea plumes after oil spills and gas leaks from seafloor. J. Loss Prev. Process Ind. pp. 1 – 11 (2017)
21. Coutinho, R.W.L., Boukerche, A., Vieira, L.F.M., Loureiro, A.A.F.: On the design of green protocols for underwater sensor networks. IEEE Commun. Mag. **54**(10), 67–73 (2016)
22. Zhou, Z., Peng, Z., Cui, J.-H., Shi, Z., Bagtzoglou, A.: Scalable localization with mobility prediction for underwater sensor networks. IEEE Trans. Mob. Comput. **10**(3), 335–348 (2011)
23. Di Rienzo, F., Girolami, M., Chessa, S., Paparella, F., Caruso, A.: Signals from the depths: properties of percolation strategies with the argo dataset. In: Proceedings of the IEEE Symposium on Computers and Communication (ISCC), pp. 372–378, June 2016
24. Stojanovic, M., Preisig, J.: Underwater acoustic communication channels: propagation models and statistical characterization. IEEE Commun. Mag. **47**(1), 84–89 (2009)
25. Brekhovskikh, L.M., Lysanov, Y.P.: Fundamentals of Ocean Acoustics. Springer (2003)
26. Stojanovic, M.: On the relationship between capacity and distance in an underwater acoustic communication channel. In: Proceedings of the 1st ACM International Workshop on Underwater Networks (WUWNet), pp. 41–47 (2006)
27. Santi, P.: Topology control in wireless ad hoc and sensor networks. ACM Comput. Surv. **37**(2), 164–194 (2005)
28. Rappaport, T.: Wireless Communications: Principles and Practice, 2nd edn. Prentice Hall PTR, Upper Saddle River, NJ, USA (2001)
29. Casari, P., Harris, A.F.: Energy-efficient reliable broadcast in underwater acoustic networks. In: Proceedings of the 2nd Workshop on Underwater Networks (WUWNet), pp. 49–56 (2007)
30. Porto, A., Stojanovic, M.: Optimizing the transmission range in an underwater acoustic network. In: Proceedings of the Oceans, pp. 1–5 (2007)
31. Su, Y., Zhu, Y., Mo, H., Cui, J.-H., Jin, Z.: A joint power control and rate adaptation mac protocol for underwater sensor networks. Ad Hoc Netw. **26**, 36–49 (2015)
32. Kredo-II, K., Djukic, P., Mohapatra, P.: Stump: exploiting position diversity in the staggered tdma underwater mac protocol. In: Proceedings of the IEEE INFOCOM, pp. 2961–2965, April 2009
33. Hsu, C.-C., Lai, K.-F., Chou, C.-F., Lin, K.C.J.: St-mac: spatial-temporal mac scheduling for underwater sensor networks. In: Proceedings of the IEEE INFOCOM, pp. 1827–1835, April 2009
34. Partan, J., Kurose, J., Levine, B.N.: A survey of practical issues in underwater networks. SIGMOBILE Mob. Comput. Commun. Rev. **11**(4), 23–33 (2007)
35. Coutinho, R.W.L., Boukerche, A., Vieira, L.F.M., Loureiro, A.A.F.: Modeling and analysis of opportunistic routing in low duty-cycle underwater sensor networks. In: Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), pp. 125–132 (2015)
36. Coutinho, R.W.L., Boukerche, A., Vieira, L.F.M., Loureiro, A.A.F.: Modeling the sleep interval effects in duty-cycled underwater sensor networks. In: Proceedings of the IEEE International Conference on Communications (ICC), pp. 1–6, May 2016
37. Coutinho, R.W.L., Boukerche, A., Vieira, L.F.M., Loureiro, A.A.F.: A novel centrality metric for topology control in underwater sensor networks. In: Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), pp. 205–212 (2016)
38. Younis, M., Senturk, I.F., Akkaya, K., Lee, S., Senel, F.: Topology management techniques for tolerating node failures in wireless sensor networks: a survey. Comput. Netw. **58**, 254–283 (2014)
39. Forero, P.A., Lapic, S.K., Wakayama, C., Zorzi, M.: Rollout algorithms for data storage- and energy-aware data retrieval using autonomous underwater vehicles. In: Proceedings of the International Conference on Underwater Networks & Systems (WUWNET), pp. 22:1–22:8 (2014)
40. Coutinho, R.W.L., Vieira, L.F.M., Loureiro, A.A.F.: DCR: depth-controlled routing protocol for underwater sensor networks. In: Proceedings of the IEEE Symposium on Computer and Communication (ISCC), pp. 453–458 (2013)

41. Coutinho, R.W.L., Vieira, L.F.M., Loureiro, A.A.F.: Movement assisted-topology control and geographic routing protocol for underwater sensor networks. In: Proceedings of the 6th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems (MSWiM), pp. 189–196 (2013)
42. Coutinho, R.W.L., Boukerche, A., Vieira, L.F.M., Loureiro, A.A.F.: Gedar: geographic and opportunistic routing protocol with depth adjustment for mobile underwater sensor networks. In: Proceedings of the IEEE International Conference on Communications (ICC), pp 251–256, June 2014
43. Coutinho, R.W.L., Boukerche, A., Vieira, L.F.M., Loureiro, A.A.F.: A novel void node recovery paradigm for long-term underwater sensor networks. Ad Hoc Netw. **34**, 144–156 (2015)
44. Coutinho, R.W.L., Boukerche, A., Vieira, L.F.M., Loureiro, A.A.F.: Geographic and opportunistic routing for underwater sensor networks. IEEE Trans. Comput. **65**(2), 548–561 (2016)
45. Coutinho, R.W.L., Boukerche, A., Vieira, L.F.M., Loureiro, A.A.F.: EnOR: energy balancing routing protocol for underwater sensor networks. In: Proceedings of the IEEE Internationl Conference on Communications (ICC), pp. 3293–3298, May 2017
46. Cayirci, E., et al.: Wireless sensor networks for underwater survelliance systems. Ad Hoc Netw. **4**(4), 431–446 (2006)
47. Erol, M., Vieira, L.F.M., Gerla, M.: Localization with dive'n'rise (dnr) beacons for underwater acoustic sensor networks. In: Proceedings of the 2nd Workshop on Underwater Networks (WuWNet), pp. 97–100 (2007)
48. Coutinho, R.W.L., Boukerche, A., Vieira, L.F.M., Loureiro, A.A.F.: Design guidelines for opportunistic routing in underwater networks. IEEE Commun. Mag. **54**(2), 40–48 (2016)
49. Xie, P., Cui, J.-H., Lao, L.: VBF: vector-based forwarding protocol for underwater sensor networks. In: Networking'06, pp. 1216–1221 (2006)
50. Nicolaou, N., See, A., Xie, P., Cui, J.-H., Maggiorini, D.: Improving the robustness of location-based routing for underwater sensor networks. In: Proceedings of the OCEANS 2007—Europe, pp. 1–6, June 2007
51. Hwang, D., Kim, D.: DFR: directional flooding-based routing protocol for underwater sensor networks. In: Proceedings of the OCEANS, pp. 1–7 (2008)
52. Yan, H., Shi, Z.J., Cui, J.-H.: DBR: depth-based routing for underwater sensor networks. In: Proceedings of the 7th International IFIP-TC6 Networking, pp. 72–86 (2008)
53. Xie, P., Zhou, Z., Peng, Z., Cui, J.-H., Shi, Z.: Void avoidance in three-dimensional mobile underwater sensor networks. Wirel. Algorithms Syst. Appl. (LNCS) **5682**, 305–314 (2009)
54. Lee, U., Wang, P., Noh, Y., Vieira, L.F.M., Gerla, M., Cui, J.-H.: Pressure routing for underwater sensor networks. In: Proceedings of the 29th Conference on Information Communications (INFOCOM), pp. 1676–1684 (2010)
55. Noh, Y., Lee, U., Wang, P., Choi, B.S.C., Gerla, M.: Vapr: Void-aware pressure routing for underwater sensor networks. IEEE Trans. Mob. Comput. **12**(5), 895–908 (2013)
56. Chirdchoo, N., Soh, W.-S., Chua, K.C.: Sector-based routing with destination location prediction for underwater mobile networks. In: Proceedings of the Internationl Conference on Advanced Information Networking and Applications Workshops (WAINA '09), pp. 1148–1153, May 2009
57. Basagni, S., Petrioli, C., Petroccia, R., Spaccin, D.: Channel-aware routing for underwater wireless networks. In: 2012 Proceedings of the Oceans, pp. 1–9. Yeosu, May 2012
58. Nowsheen, N., Karmakar, G., Kamruzzaman, J.: An opportunistic message forwarding protocol for underwater acoustic sensor networks. In: Proceedings of the Asia-Pacific Conference on Communications (APCC), pp. 172–177, Aug 2013
59. Carlson, E.A., Beaujean, P.-P., An, E.: Location-aware routing protocol for underwater acoustic networks. In: Proceedings of the Oceans, pp. 1–6 (2006)
60. Chen, Y.-D., Chen, Y.-W., Lien, C.-Y., Shih, K.-P.: A channel-aware depth-adaptive routing protocol for underwater acoustic sensor networks. In: 2014 Proceedings of the Oceans, pp. 1–6, Taipei, April 2014
61. Chen, Y.-D., Lien, C.-Y., Wang, C.-H., Shih, K.-P.: DARP: a depth adaptive routing protocol for large-scale underwater acoustic sensor networks. In: Proceedings of the IEEE Oceans, , pp. 1–6 (2012)

62. Chen, J., Wu, X., Chen, G.: REBAR: a reliable and energy balanced routing algorithm for uwsns. In: Proceedings of the International Conference on Grid and Cooperative Computing (GCC), pp. 349–355, Oct 2008
63. Wahid, A., Lee, S., Jeong, H.-J., Kim, D.: EEDBR: energy-efficient depth-based routing protocol for underwater wireless sensor networks. Adv. Comput. Sci. Inf. Technol. **195**, 223–234 (2011)
64. Hu, T., Fei, Y.: QELAR: a machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks. IEEE Trans. Mob. Comput. **9**(6), 796–809 (2010)
65. Miquel Jornet, J., Stojanovic, M., Zorzi, M.: Focused beam routing protocol for underwater acoustic networks. In: Proceedings of the 3rd ACM International Workshop on Underwater Networks (WUWNet), pp. 75–82 (2008)
66. Coutinho, R.W.L., Boukerche, A., Vieira, L.F.M., Loureiro, A.A.F.: Performance modeling and analysis of void-handling methodologies in underwater wireless sensor networks. Comput. Netw. **126**, 1–14 (2017)
67. Ghoreyshi, S.M., Shahrabi, A., Boutaleb, T.: Void-handling techniques for routing protocols in underwater sensor networks: survey and challenges. IEEE Commun. Surv. Tutor. **19**(2), 800–827, Secondquarter 2017
68. Coutinho, R.W.L., Boukerche, A., Vieira, L.F.M., Loureiro, A.A.F.: Local maximum routing recovery in underwater sensor networks: performance and trade-offs. In: Proceedings of the IEEE 22nd International Symposium on Modelling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTs), pp. 112–119, Sept 2014
69. Jaffe, J., Schurgers, C.: Sensor networks of freely drifting autonomous underwater explorers. In: Proceedings of the 1st ACM International Workshop on Underwater Networks (WUWNet), pp. 93–96 (2006)
70. Biswas, S., Morris, R.: Exor: opportunistic multi-hop routing for wireless networks. SIGCOMM Comput. Commun. Rev. **35**(4), 133–144 (2005)

Geometric Routing Without Coordinates but Measurements

Pierre Leone and Kasun Samarasinghe

Abstract Geometric routing is a routing scheme proposed for networks with highly dynamic topology, like wireless ad hoc networks. It uses the geometric coordinates of the nodes and makes routing decisions based on the geometric properties between them. Hence, it does not build and maintain routing tables. Nevertheless, as geometric routing requires an auxiliary location service to equip nodes with geometric location, there are not much deployments in real network settings. In this chapter, we investigate an efficient localization system called Virtual Raw Anchor Coordinates (VRACs), which is an anchor-based coordinate system. It assigns the raw distances from anchors as the coordinates of nodes, hence avoiding further computations. Despite its efficiency, it is not possible to perform geometric operations on VRAC. In this manuscript, we propose alternative constructs to perform geometric routing over VRAC. Initially, a greedy routing algorithm is developed, which is then combined with a face routing strategy, when greedy routing does not guarantee delivery. Moreover, we present the conditions over which greedy routing can be performed on VRAC. In Sect. 5, a geometric routing algorithm is presented, where greedy and face routing are combined to guarantee the delivery of messages. In Sect. 6, a greedy routing algorithm is presented and proved to be successful given that certain connectivity conditions are satisfied.

1 Introduction—Routing in Wireless Ad Hoc Networks

Advancements in hardware technologies have made the communication and computation more ubiquitous. Not only with widely available handheld devices with high-end processors but also with various tiny artifacts with communication capabilities have given rise to the computing paradigm called ubiquitous computing.

P. Leone (✉) · K. Samarasinghe
Department of Computer Science, University of Geneva, Geneva, Switzerland
e-mail: pierre.leone@unige.ch

K. Samarasinghe
e-mail: kasun.wijesiriwardana@unige.ch

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_16

It opens up the possibility for plethora of applications in various domains, ultimately enhancing the communication and computation experience of the end users.

Wireless ad hoc network is a class of networks, which plays a major role in providing the underlying communication infrastructure for ubiquitous systems. These networks are comprised of devices connected over the wireless medium, without a predefined infrastructure. Devices within the wireless communication range of each other connect and establish a link between them. The communication between two such devices is referred to as a single-hop communication. The overall network is the collection of these communication links, which forms a wireless mesh between the devices. Communication between two devices which are not in each one's vicinity has to be done in multiple hops.

The connectivity of the network is subject to various different factors such as wireless hardware of the devices and the quality of wireless links between the devices. In turn, quality of wireless links depends upon radio communication characteristics like interference and multi-path fading. As a result, wireless links can be disrupted and re-established over time. Hence not only the connectivity mesh is ad hoc, but also it changes over time. Typically, the devices used in wireless ad hoc networks are constrained by computation and power supply. Thus, the operating system as well as the applications have to consider these limitations in their design. Despite the challenging nature of wireless communication, wireless ad hoc networks offer a great flexibility in terms of the deployment. This leads to various applications of wireless ad hoc networks in various domains. A classical example of a wireless ad hoc network is environmental monitoring, where a set of wireless devices with sensory capabilities are deployed to gather real-time environmental data. Another application is to augment the environment with devices to create smart environments with certain capabilities to improve the quality of life of the inhabitants.

Design of routing protocols for wireless ad hoc networks is a challenging task, due to the dynamic nature of the topology. Obviously, it is not feasible to apply the design of classical network protocols, where it gathers the topology knowledge by flooding the network and determines the route by computing the shortest path between two network nodes. One of the alternative paradigms for dynamic networks is on-demand routing where the routes are discovered upon the routing request. Ad hoc On-Demand Vector (AODV) routing [1] is a seminal work on this line of research. In AODV, routes are discovered per routing request. It broadcasts a route request along the network until it reaches the destination. On the way back, nodes which pass the message store the route in their local routing table. This approach obviously solves the problem of pre-computed routing tables as well as it discovers the routes dynamically. A similar alternative routing scheme for wireless ad hoc networks is Optimized Link State Routing (OLSR), which is a proactive link state routing protocol. It performs an optimized flooding to gather the topology knowledge, so that the routes can be computed.

Another alternative paradigm is hierarchical routing, where routing is performed in a hierarchical structure of the network. Cluster-based routing is a hierarchical routing scheme proposed in the general context of wireless ad hoc networks. It partitions the network into clusters and constructs a routing structure between the clusters.

Clustering is a well-studied problem, where there are various schemes proposed for different networking scenarios. A generic cluster-based routing scheme works in two layers. In the bottom layer, routing can be performed based on a tree-like structure, where all the nodes in the cluster include in a tree rooted at the cluster head (can be arbitrary node). In the top level, a routing scheme has to be maintained between the clusters to support intercluster routing. A straightforward choice is to perform a shortest path algorithm between the cluster level and maintain a routing table. This approach is proposed in [2], where they perform classical link state routing in cluster level. Link state protocol needs to acquire the complete network topology knowledge by flooding the network between the clusters.

1.1 Local and Stateless Routing

In a routing algorithm, it is important to quantify the amount of knowledge on the communication graph required by a routing algorithm. When the algorithm has the complete knowledge of the communication graph, it is considered to be a global algorithm. A local algorithm makes routing decision only based on the local knowledge of a node. In another perspective, it suggests to perform routing only by use of local neighborhood information, which is denoted as $\mathcal{N}_u$. Accordingly, a local algorithm (or 1-local algorithm) executes by only using the *1-hop* neighborhood information of the network graph. Therefore, algorithms which require *k-hop* neighborhood information, where k is a constant, can still be considered as local algorithms. Nevertheless, when k grows with the number of nodes in the network ($k = O(n)$) it can no longer be called as a local algorithm. An important restriction on local routing algorithms was presented in [3], which is summarized in Theorem 1.

Theorem 1 *[3] For every $k < \lfloor n/2 \rfloor$, every k-local routing algorithm fails on some connected graph.*

Therefore given an arbitrarily connected graph, it is not possible to route only with a constant neighborhood information. Nevertheless, if the given graph is a geometric graph and certain connectivity conditions are met, local routing algorithms exist.

Another important quantification of a routing algorithm is the routing state (information stored in the memory) maintained by a routing algorithm. The routing state can be in two forms, as the state stored at a node and the state maintained in the message header. A given node can maintain details of routes like in AODV [1] or maintain a distributed routing structure like a spanning tree. For instance, in a classical routing protocol like RIP, the state maintained at a node is $\mathcal{O}(n)$. Since a message header is limited, ideally it should carry a constant number of node information.

In terms of the scalability of a routing protocol, it is important to consider both locality and the routing state maintained. In particular, for large-scale ad hoc networks, a routing protocol has to trade-off between these two quantities in order to achieve scalability.

1.2 Geometric Routing

Geometric routing [4, 5] is a routing paradigm, specially designed for wireless ad hoc networks. It avoids constructing and maintaining routing tables and instead uses the geographic coordinates of nodes to make routing decisions. To fix ideas, each node of the network may be given its geographical coordinates with the help of electronic device like a Global Positioning System (GPS). The position plays the role of the address in classical routing schemes. However, a sending node, given the geographical position of the destination node of a message, its position, and the ones of neighboring nodes, can decide the node to which it forwards the message. For instance, it forwards the message to the neighboring node that is the closest to the destination with respect to the Euclidean distance; see Sects. 1.2.1 and 1.2.2.

Such routing schemes are called *implicit* since the addresses provide all the routing information [6]. In contrast, *explicit* routing schemes, like distance vector routing protocol, or data gathering protocols, like collection tree protocol [7], require that nodes compute and maintain routing information. Usually, a routing table that consists in $\mathcal{O}(n)$ entries, where n is the number of nodes in the network, must be maintained. This table associates to each possible destination address the suitable neighboring node to which it forwards the message.

Therefore, geographic routing keeps the communication overheads minimal, in turn saving energy consumption of the nodes. There are two phases in geometric routing, namely *greedy routing* and *face routing*. It is important to note that, geometric routing requires the nodes to be embedded in a geometric surface.

1.2.1 Greedy Routing

Greedy routing (also referred to as greedy forwarding) is the simplest form of geometric routing. It makes a local *greedy decision*, when forwarding a packet, simply choosing the geographically closest neighbor toward the destination as the successor. Let S be the source node and D be the destination node. At a given node u starting from S, it decides the next node v to which the packet has to be forwarded according to the following criterion. $\mathcal{N}_u$ is the set of *1-hop* neighbors of u, and $d(.)$ is the underlying distance function in the geometric surface.

$$v = \underset{v \in \mathcal{N}_u}{\operatorname{argmin}}\ d(v, D) \tag{1}$$

Depending on the graph embedding, distance function can be either Euclidean or non-Euclidean. These distance functions (or metrics) usually need to follow the metric criteria. Even more abstract conditions on a distance function for greedy routing were proposed in [8]. It showed that in order to form a greedy path, following conditions must be satisfied by the distance function.

1. Transitivity: If node y is greedy for x and z is greedy for y, then z is greedy for x.
2. Anti-symmetry: If y is greedy for x, then x is not for y.

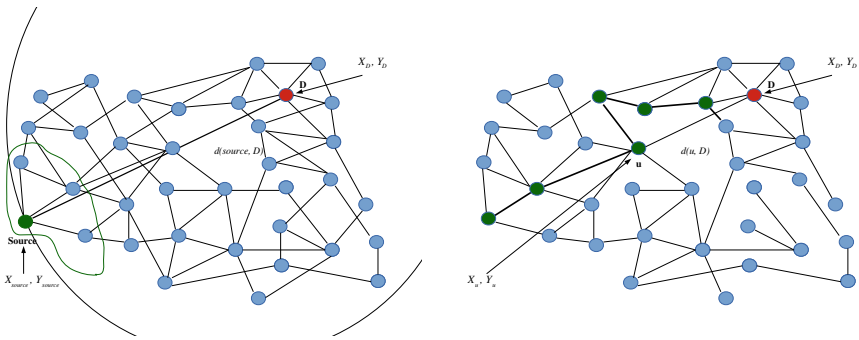
These properties can be used to formulate a greedy routing algorithm when metric functions are not available (see Sects. 5.1 and 6).

1.2.2 Face Routing

Greedy routing may be stuck in a node, where greedy routing is no longer possible. Such nodes are referred to as *local minima* or *routing voids*. These nodes are unavoidable in wireless ad hoc network deployments. In order to recover the routing process from a local minima, GFG [4] and GPSR [5] introduced the concept of face routing. Face routing is a systematic exploration of the graph, which guarantees either to reach a node where greedy routing can be resumed or to reach the destination.

Face routing requires the underlying communication graph to be a planar graph. It uses the classical maze-solving strategy called *left/right-hand rule*. Initially it applies that rule and explores the current face, where greedy routing got stuck. Along the exploration, it performs a face switching, which guarantees a progress toward the destination (see Fig. 1b). Face routing guarantees the delivery of a message. A comprehensive analysis of the guarantee of delivery of different face switching strategies is discussed in [9]. The strategy used in GFG is successful in arbitrary planar subgraph. Even though face routing guarantees delivery, due to exhaustive exploration of the faces of the graph, routing stretch can be arbitrarily high. A worst-case optimal face routing strategy was introduced in [10] assuming a unit disk communication graph model.

In a practical perspective, a local planarization of the communication graph is challenging. In GFG and GPSR, they obtain a planar subgraph assuming an ideal radio communication characteristics (*unit disk graph*). Each node drops links in the routing process, which could lead to crossing edges (hence non-planar) obtaining



(a) Greedy Routing (b) Face Routing

Fig. 1 Two phases of geometric routing

a planar subgraph. More specifically, in GPSR, nodes locally apply a condition for planarity introduced in [11]. Relative neighborhood graph [12] is also an alternative to gabrial graph, where its properties can be used to extract a connected planar subgraph.

Nevertheless, by the time of this writing, there is no planarization algorithm (which is local) known for arbitrarily connected graphs. An alternative proposal is to avoid the planarization [13] and instead maintain a distributed data structure to cover the void region. This data structure is then used to overcome the dead end in greedy forwarding. As this approach maintains a certain amount of state across some nodes in the network, it does not meet the original goal of geographic routing.

2 Geometric Routing on Virtual Raw Anchor Coordinates

In this chapter, geometric routing algorithms on a coordinate system called Virtual Raw Anchor Coordinates (VRACs) is presented. VRAC is an efficient coordinate system devised for geometric routing in wireless ad hoc and sensor networks. It is important to emphasize that algorithms presented are local algorithms, hence given a VRAC deployment no additional overhead is incurred. In Sect. 3, construction of VRAC and related definitions are presented, followed by the planar graph construction on VRAC in Sect. 4. In Sect. 5, a geometric routing algorithm is presented, where greedy and face routing are combined to guarantee the delivery. In Sect. 6, a greedy routing algorithm is presented, when the graph connectivity follows certain conditions.

2.1 Why Another Coordinate System?

Localization of nodes in wireless ad hoc networks is not a trivial task. Hardware-dependent solutions, where a set of anchor nodes have to be designated, require human intervention and incur higher initialization costs. Furthermore, such solutions only work in outdoor environments, as the anchor nodes are equipped with GPS hardware. The alternative paradigm of virtual coordinates is inherently computationally intensive, since most of these mechanisms require iterative computations. As a result, associated communication overhead is overwhelming especially in large-scale networks.

Identifying these discrepancies, GLIDER [14] and BVR [15] have independently proposed an anchor-based virtual coordinate scheme, specially for geometric routing. Given a pre-designated set of anchor nodes, both these protocols assign coordinates to the nodes, simply as a hop-count vector from the anchors. Therefore, it does not require any further computation or complicated communication between nodes.

Virtual Raw Anchor Coordinate (VRAC) system is a coordinate system specially defined for geometric routing. Coordinate construction in VRAC is conceptually

similar to GLIDER and BVR, where it assigns raw distances from anchors as the coordinates. In contrast to GLIDER and BVR, it uses the Euclidean distance from anchor nodes, instead of the hop distance.

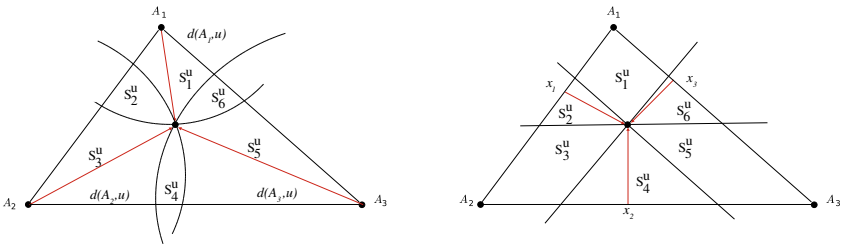
3 Virtual Raw Anchor Coordinate System

In order to define the VRAC system, consider a network deployed along with three anchor nodes A_1, A_2, and A_3. If the nodes are distributed within the triangle formed by three anchors $\widehat{A_1A_2A_3}$, coordinates are defined as follows.

Definition 1 A node u is assigned $\big(u_1, u_2, u_3\big) = \big(d(u, A_1), d(u, A_2), d(u, A_3)\big)$, where $d(u, A_i)$ is the Euclidean distance from the anchor A_i.

In this chapter, two versions of VRAC constructions are investigated. First version is the basic construction of coordinates according to the Definition 1 (see Fig. 2a). On this version, algorithms are based on the combinatorial structure of the coordinate system [16], which is presented in Sect. 5. In the second version, we further assume that anchors can estimate the distances between them, hence the lengths of the sides of the triangle $\widehat{A_1A_2A_3}$ are known to each node. This leads to a slightly different version of VRAC (see Fig. 2b), where perpendicular distances from triangle edges are assigned as coordinates. Algorithms on this coordinate system are presented in Sect. 5.3.

Virtual Raw Anchor Coordinate space by definition purely is a virtual coordinate system, which does not correspond to the physical coordinates. Moreover, there is no *metric* associated in this coordinate system, hence it is not possible to perform the geometric computations required by geometric routing. This motivates us to explore combinatorial properties of VRAC which can be used in geographic routing.
Inspired by Schnyder characterization of planar graphs [17], we can define three order relations on the set of nodes V.



(a) Basic coordinate assignments with raw distances from anchors

(b) Coordinate assignment with perpendicular distances from edges of the triangles

Fig. 2 Two VRAC variants

Definition 2 The three order relations $<_i$, $i = 1, 2, 3$ on $V \times V$ are defined by

$$\forall u, v \in V \quad u <_i v \iff d(u, A_i) > d(v, A_i) \iff u_i > v_i.$$

The three order relations are total,[1] hence it is possible to define the minimum of a set with respect to the three orders. We will denote this by $\min_i$ for $i = 1, 2, 3$. These three orders permit the definition of sectors associated with a node u.

Definition 3 We define the following sectors associated to a node $u \in V$; see Fig. 2a. Note that the reference node u does not belong to the sectors.

$$\begin{aligned}
s_1^u &= \{v \mid u <_1 v,\ u >_2 v,\ u >_3 v\} \cap \widehat{A_1A_2A_3}.\\
s_2^u &= \{v \mid u <_1 v,\ u <_2 v,\ u >_3 v\} \cap \widehat{A_1A_2A_3}.\\
s_3^u &= \{v \mid u >_1 v,\ u <_2 v,\ u >_3 v\} \cap \widehat{A_1A_2A_3}.\\
s_4^u &= \{v \mid u >_1 v,\ u <_2 v,\ u <_3 v\} \cap \widehat{A_1A_2A_3}.\\
s_5^u &= \{v \mid u >_1 v,\ u >_2 v,\ u <_3 v\} \cap \widehat{A_1A_2A_3}.\\
s_6^u &= \{v \mid u <_1 v,\ u >_2 v,\ u <_3 v\} \cap \widehat{A_1A_2A_3}.
\end{aligned}$$

In the following, we refer to these sectors as sector number $1, 2, \ldots, 6$, respectively, i.e., for instance, s_4^u is sector number 4 of u. Notice that the nodes in the network are assumed to belong to the interior of the triangular region $\widehat{A_1A_2A_3}$, defined by the three anchors. This is necessary to the application of Schnyder's planarity criterion [17] that we use to extract a planar subgraph of the communication graph (see conditions in Eq. (2) and Definition 5).

Definition 4 Given a node D, we also use the convenient notation s_D^u to denote the sector j of u such that $D \in s_j^u$, i.e., $D \in s_D^u$.

4 Graph Planarization on Virtual Raw Anchor Coordinate System

For combined greedy–face routing, a planar subgraph of the communication graph has to be extracted. Given the VRAC coordinates, it is not possible to extract a gabriel graph [11] or relative neighborhood graph [12]. Nevertheless, with the combinatorial properties available in VRAC, it is possible to use the classical Schnyder characterization [17] defined below to formalize a planarization algorithm. Given a planar graph $G = (V, E)$, it is proven in [17] that there exist three total order relations on $V \times V$, denoted $<_1, <_2, <_3$ such that;

[1] A total order is a binary relation which is valid for *all the pairs* in a set.

$$\begin{aligned} &a)\ \ \bigcap_{i=1,2,3} <_i = \emptyset, \text{ and} \\ &b)\ \ \forall (x,y) \in E, \forall z \notin \{x,y\}\ \exists i \in \{1,2,3\} \\ &\text{s.t. } x <_i z \text{ and } y <_i z. \end{aligned} \tag{2}$$

This is called a (*three-dimensional*) *representation* of a planar graph. Such a representation of a planar graph does not use a (planar) embedding and applies to an abstract graph. Based on this characterization, Huc et al. [18] defined a planar subgraph on the VRAC system as follows, in order to formulate a local planarization algorithm.

Definition 5 Given a graph $G(V, E)$, a planar subgraph $\widetilde{G}(\widetilde{V}, \widetilde{E})$ can be defined such as

1. $\widetilde{V} = V$
2. $\widetilde{E} = \left\{ (u,v) \,\middle|\, v \in s^u_{2k-1} \text{ and } v = \min_k(s^u_{2k-1})\ k = 1, 2 \text{ or } 3 \text{ and } (u,v) \in E \right\}$

According to the definition, if a node can locally determine the minimum edge with respect to the order relation (in turn, with respect to the sector), it is possible to planarize the graph. Huc et al. [18] showed that this is possible if the connectivity graph follows the unit disk graph assumption. Thereby, they formulate a local planarization algorithm, where a node keeps the minimum edge and ignore all the other edges, resulting in a planar subgraph.

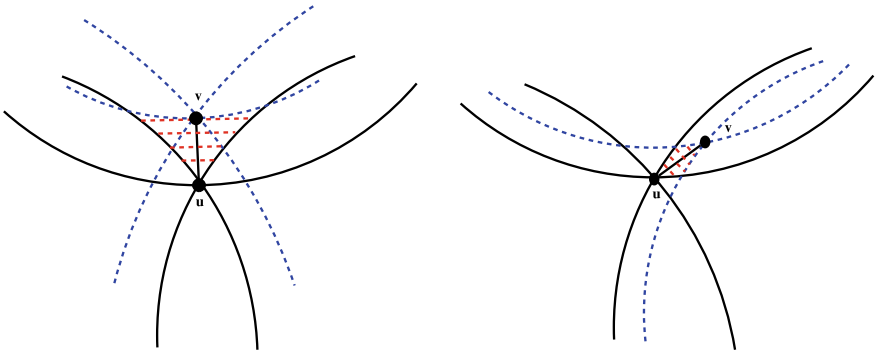
However, a node u can have many neighboring nodes in the sectors $v \in s^u_2, s^u_4$ or $v \in s^u_6$. In [18], we call the edges (u, v) with v in s^u_1, s^u_3, or s^u_5 *outgoing* edges. With this terminology, a node has at most three outgoing edges and possibly many *ingoing* edges (and edges (u, v) such that $v \in s^u_2, s^u_4$ or $v \in s^u_6$). We emphasize that this is a useful denomination but the graph $\widetilde{G}$ is not oriented. There are several important properties of the obtained planar graph following the Definition 5, which are used in the formulation of routing algorithms throughout the rest of this chapter.

Property 1 *A node u has at most one neighboring node in each of* s^u_1, s^u_3, s^u_5 *(the closest with respect to the corresponding order relation).*

This follows immediately from the Definition 5. Indeed, if $v \in s^u_1$, $v \in s^u_3$, or $v \in s^u_5$ then $u \in s^v_4$, $u \in s^v_6$, or $u \in s^v_2$, respectively, and then, the only possibility for an edge $(u, v) \in \widetilde{E}$ is that v is minimal with respect to $<_1$, $<_3$ or, $<_5$, respectively. Due to the elimination of edges in the planarization process, empty regions around an edge can be observed.

Property 2 *Given that there is an edge between node u and v and* $v \in s^u_{2i-1}$*, then there are no other nodes in the region defined by;*

$$s^u_{2i-1} \cap \{z \mid d(A_i, z) > d(A_i, v)\}$$



(a) Illustration of property 2 (b) Illustration of property 3

Fig. 3 Empty regions around an edge (u, v)

Proof By the construction of the planar graph, v is the minimal node in s^u_{2i-1}, with respect to the order relation $<_i$ (defined by $d(A_i, z)$). Therefore to ensure planarity, by definition there is no $w \in s^u_{2i-1}$ such that $d(A_i, w) > d(A_i, v)$ or similarly $w <_i v$ (see Fig. 3a).

An immediate implication of Property 2 is an even finer empty region illustrated in Fig. 3b as presented in Property 3.

Property 3 *Given the nodes u and v such that $(u, v) \in \widetilde{E}$, then the region defined by $s^u_v \cap s^v_u = \emptyset$*

Proof Without loss of generality, we can assume that $u >_1 v$, $u >_2 v$, $u <_3 v$, $v \in s^u_5$ (the proof is the same if we permute the indices). Because u and v are connected, it must be that $v = min_3\{z \mid u >_1 z, u >_2 z, u <_3 z\}$. Then sector s^v_u is defined by $\{z \mid z >_1 v, z >_2 v, z <_3 v\}$, and the intersection is $s^u_v \cap s^v_u = \{z \mid u >_1 z >_1 v, u >_2 z >_2 v, u >_3 z <_3 v\}$. Hence, if the intersection $s^u_v \cap s^v_u$ is not empty, u should be connected to a node inside $s^u_v \cap s^v_u$ instead of v.

As per these properties, if there is an edge (u, v) in $\widetilde{E}$ and $v \in s^u_{2i-1}$, then there are no nodes $w \in s^u_{2i-1}$ such that $w <_i v$ and $(u, w) \notin E$; i.e., the node w should be connected to u to ensure planarity. A local algorithm has to overcome a pathological situation which occurs due to the radio communication irregularities, whereas a node w exists but residing out of the range of u. In such a situation, u erroneously keeps a longer edge. Huc et al. [18] proved that this situation can be overcome if the planarization algorithm has two-hop neighborhood knowledge.

5 Combined Greedy–Face Routing with Delivery Guarantees

In this section, a combined greedy–face routing algorithm on VRAC system is presented. This algorithm is a variant of the combined greedy–face routing algorithm [4]. However, the main difference is that in VRAC, it is not possible to compare the angles. As a result, it is not possible to implement face routing independent of greedy routing. In fact, the routing algorithm that we designed only uses the order relations defined in Definition 2. The distances from the anchor nodes are a convenient way of computing these order relations. Nevertheless, any order relation satisfying the conditions as in Eq. 2 is suitable. Therefore, our routing scheme can be qualified as a combinatorial scheme.

In face routing, basic primitives are the implementation of the left- or right-hand traversal rule to explore a face of the planar graph and the detection of the intersection of an edge of the path with the source destination line (see Sect. 1.2.2). Nevertheless, in VRAC it is not possible to detect line segment intersection. Therefore, it is required to consider a (*greedy*) region that contains the source and destination nodes (see Eq. (3)) and to detect when the routing path crosses this region. If the packet reaches a node in the greedy region, it switches back to greedy routing.

5.1 Greedy Routing Primitives

As mentioned earlier, VRAC does not have an underlying metric. Hence, it is not possible to decide the greedy neighbors as in the Euclidean space based on the distance function. Instead, an abstract characterization of a greedy path [8] is used to define greedy routing criteria. Accordingly, a greedy path on VRAC is defined as below.

Definition 6 For destination node D, a path $\{u^i\}_{i=1,\ldots,k}$ is a greedy path if

$$u^{i+1} \in s_D^{u^i} \bigcap s_{u^i}^D. \tag{3}$$

The region defined by the intersection $s_D^{u^i} \bigcap s_{u^i}^D$ is considered to be the *greedy region* of the node u^i with respect to the destination D. Moreover, u^{i+1} is said to be greedy for u^i with respect to D. When considering a greedy path established according to the above definition, it ensures the *transitivity* property, which is a fundamental property of a greedy path [8]. This is presented in Proposition 1, which is then used to prove the convergence of a greedy path in Corollary 1.

Proposition 1 (transitivity) *If u^{i+1} is in the greedy region of u and u^{i+2} is in the greedy region of u^{i+1}, then u^{i+2} is in the greedy region of u.*

Proof What has to be proven is that, if $u^{i+1} \in s_D^{u^i} \bigcap s_{u^i}^D$ and $u^{i+2} \in s_D^{u^{i+1}} \bigcap s_{u^{i+1}}^D$ then $u^{i+2} \in s_D^{u^i} \bigcap s_{u^i}^D$. For concreteness, we consider $s_D^{u^i} = s_5^{u^i} = \{z \mid z <_1 u^i, z <_2 u^i, z >_3 u^i\}$ and, then $s_{u^i}^D = \{z \mid z >_1 D, z >_2 D, z <_3 D\}$ (we reverse the signs of the inequalities). The assumption $u^{i+1} \in s_D^{u^i} \bigcap s_{u^i}^D$ leads to $D <_1 u^{i+1} <_1 u^i$, $D <_2 u^{i+1} <_2 u^i$, $D >_3 u^{i+1} >_3 u^i$. We then conclude that $s_D^{u^{i+1}} = \{z \mid z <_1 u^{i+1}, z <_2 u^{i+1}, z >_3 u^{i+1}\} \subset s_D^{u^i}$ and that $s_{u^i}^D = s_{u^{i+1}}^D$. This proves the proposition.

Algorithm 1 contains a pseudocode of the implementation of the greedy routing primitive.

Algorithm 1 Implementation of the routine *greedy(u,D)*

```
1: procedure GREEDY(u,D)
2:     Determine the sector s_D^u
3:     Determine the sector s_u^D                ▷ by reversing the signs of the inequalities
4:     if N_u ∩ s_D^u ∩ s_u^D ≠ ∅ then
5:         select arbitrarily x in the set
6:         return x
7:     else
8:         return No Greedy Neighbors
9:     end if
10: end procedure
```

Corollary 1 *Given a destination node D, a greedy path* $\{u^i\}$ *eventually reaches the destination D.*

Proof Applying Proposotion 1 inductively proves that $s_{u^i}^D = s_{u^j}^D$ for all nodes in the greedy path and that $D \in \cap_{i=1\ldots k} s_D^{u^i}$. Because the area of this last intersection decreases, it must eventually hold that the destination D is reached (there cannot be an infinite number of nodes in an infinitesimally small surface).

Notice that this result follows directly from the results in [8], since the paths satisfy the required axioms. However, we provide a simple and independent proof on the convergence of a greedy path in Corollary 1.

5.2 Face Routing Primitives: Combinatorial Approach

In this subsection, a face routing algorithm over VRAC is presented. This algorithm combined with greedy routing leads to a geometric routing algorithm with guaranteed delivery. Let u be the source and D the destination of a packet, then the routing algorithm at u switches to face routing if $\mathcal{N}_u \cap s_D^u \cap s_u^D = \emptyset$. The face routing algorithm selects the node v such that $v \in \mathcal{N}_u$ and the edge (u, v) is the first edge encountered when the line uD is rotated counterclockwise. The face traversal stops if the path

goes through a node belonging to $s_D^u \cap s_u^D$ or if an edge (v, w) intersects this region. In the former case, greedy routing is restarted, while in the later case it decides to switch the face accordingly (see Sect. 5.2.1). As we implement only the primitives for the left-hand traversal rule (see Algorithm 5), face switching is done if the path traverses the region (by selecting w as the next node) inverting the order of the nodes (v, w); i.e., the node v continues the execution of the left-hand traversal algorithm by assuming that the data is received from node w.

Notice that our implementation is not an implementation of a classical algorithm like GFG or GPSR [4, 5]. Our version of face routing cannot be executed without greedy routing. We implement the *left-hand rule* by determining the first edge encounter in the *counterclockwise* direction from an edge or the respective empty region (see Algorithm 5). Moreover, we use the face switching strategy used in GFG, which guarantees delivery on an arbitrary planar graph [9]. However, our implementation follows the rule of GFG for face switching.

Given an edge (u, v), a first primitive to implement the face traversal is to rotate the edge around u counterclockwise and to determine the next edge (u, w) that we encounter (see Algorithm 5). Actually, this amounts to find the edge (u, w) that makes the smaller angle with (u, v) where the angle is measured counterclockwise. In our coordinate system, we cannot compute the angles since we only know the order relations defined by the three anchors.

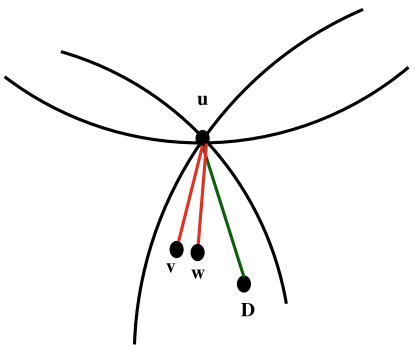
Similar to greedy routing, face routing primitives are not straightforward to implement over VRAC. In fact, given only the VRAC and initial sectoring, it is not possible to distinguish the edge positions, as required by face routing. For instance, it is not possible to determine the first edge encounters counterclockwise direction from uD line, if there is no edge between u and D, which is presented in Proposition 2.

Proposition 2 *If the nodes $v, w \in s_D^u$ and, $D \notin \mathcal{N}_u$ while $v, w \in \mathcal{N}_u$ it is not possible on VRAC to determine which edge (u, v) or (u, w) is the next edge to the line uD.*

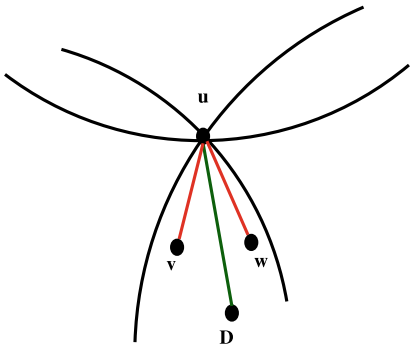
Proof Figure 4 shows two configurations where $v, w, D \in s_4^u$ and where the same order relations exist between the nodes, i.e., $D <_1 v$, $w <_1 u$, $D >_2 v$, $w >_2 u$, and $D >_3 v$, $w >_3 u$. We observe on the figure that it is not possible to determine which of (u, v) or (u, w) is next to uD.

We emphasize that this impossibility is due to the fact that uD is not an edge in the graph, but a hypothetical line between the destination and the current node (see Fig. 1b). If uD corresponds to an edge, we could investigate the empty region properties (Property 2 and Property 3) to determine the orientation of an edge around node. In fact, considering these properties, we define an ordering scheme for edges around a given node. First, a local ordering criterion is derived within a sector (see Proposotion 3), which is used to build a global ordering of edges around a node.

Local ordering considers the counterclockwise ordering and determines the rightmost node within a sector (more precisely in sectors 2, 4, and 6). Proposition 3 presents a procedure to determine the rightmost edge among two nodes v and w with respect to u, which is denoted as $right(u, v, w)$,

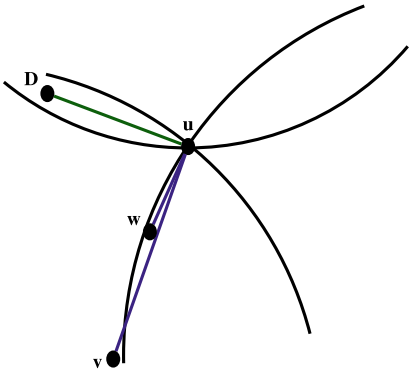


(a) Both edges in the same side of the *uD* line

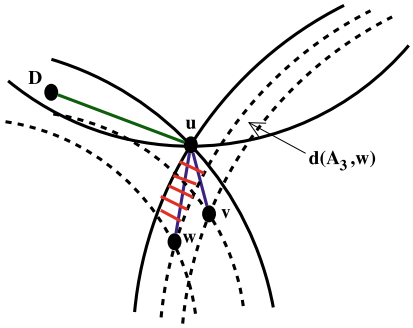


(b) Two edges in different sides of the *uD* line

Fig. 4 Illustration of the proof of Proposition 2



(a) Illustration of the impossibility in the proof of Proposition 3, where it is impossible to have a v as in the Fig. with the presence of w



(b) An illustration that the by comparing $d(A_3, v)$ and $d(A_3, w)$ it is possible to determine the edge that w is on the right of side of v

Fig. 5 Illustration of implementation of right function

Proposition 3 *Given a node u, let v and w be two nodes such that* $(u, v), (u, w) \in \widetilde{E}$ *and* $s_v^u = s_w^u = s_{2i}^u$*, respectively, then the rightmost edge is determined by;*

$$\textbf{\textit{right}}(u, v, w) = \begin{cases} v & \text{if } d(A_{1+i \bmod 3}, v) < d(A_{1+i \bmod 3}, w) \\ w & \text{otherwise} \end{cases} \tag{4}$$

Proof Let us assume that $v, w \in s_4^u$ (proofs for other sectors follow the same argument). Notice that it is not possible that u and v belong to s_1^u, s_3^u or s_5^u, as there can be only one edge in these sectors by Property 1. First we show that the impossibility of an unfavorable setting illustrated in Fig. 5a. In fact, if w were in the mentioned region

Algorithm 2 Implementation of the routine *right(u,v,w)*

```
1: procedure RIGHT(u,v,w)                ▷ v, w ∈ N_u and v, w belong to the same sector of u
2:     if v, w ∈ s_2^u then
3:         if d(A_2, w) < d(A_2, v) then return v
4:         else return w
5:         end if
6:     end if
7:     if v, w ∈ s_4^u then
8:         if d(A_3, w) < d(A_3, v) then return v
9:         else return w
10:        end if
11:    end if
12:    if v, w ∈ s_6^u then
13:        if d(A_1, w) < d(A_1, v) then return v;
14:        else return w;
15:        end if
16:    end if
17: end procedure
```

then $v \in s_1^w$, like u. But, because $u >_1 v >_1 w$ then w would be connected to v instead of u, leading to a contradiction. In order to prove the main result, following argument suffices. As there is an edge (u, w), the node v cannot be in the region highlighted in red lines in Fig. 5b (see Property 3). As this holds for v reciprocally, it is possible to determine the edge to the right by comparing $d(A_3, w)$ and $d(A_3, v)$ accordingly.

An implementation of the procedure *right*(u, v, w) is presented in pseudocode in Algorithm 2. Note that it compares the respective orders in sectors 2, 4, and 6 accordingly. Once the local ordering in each sector can be determined with the procedure *right*(u, v, w), it can be used to define a global numbering. It comes as an extension to the initial sector numbers (see Definition 3). Extended numbering is a relative scheme, where numbers are assigned to a sector relative to the sector s in two possible settings as follows.

1. s = s_D^u when $(u, D) \notin \widetilde{E}$
2. s = s_v^u, when u receives a message from v.

First setting corresponds to the scenario, where it switches from greedy routing to face routing as illustrated in Fig. 6a. In this setting, since there are no neighboring nodes in the greedy region, we need to consider the first edge encounters in the counterclockwise direction from the greedy region. Second setting corresponds to the scenario where face routing is continued as illustrated in Fig. 6b. Hence, it has to consider the first edge encounters in the counterclockwise direction with respect to the incoming edge.

We need to define the global numbering such that it determines the first edge situated in the counterclockwise direction with respect to the above two cases. It is trivial to observe that the initial sector numbers make sure such an ordering (coupled

(a) Starting face routing when $D \in s_i^u$ such that$i = 1,3,5$, i.e $s_X^u \cap s_u^X \cap \mathcal{N}_u = \emptyset$, v is on the left of uX

(b) Starting or continuing face routing when either D *or* $v \in s_i^u$ such that $i = 2,4,6$, v is on the left of uX

Fig. 6 Illustration of two settings considered in left function

with local ordering), except for the sector s. In sector s, it is possible to distinguish the edges, which are left to the uD line or (u, v) edge due to the empty regions around an edge in the planarized graph (see Propositions 2 and 3). Following proposition determines (denoted as $left(u, X, v)$) if a given edge $(u, v) \in \widetilde{E}$ is in the left (or right) of the line drawn between u and X. Note that this is a generalization of the above two settings, hence (u, X) may or may not correspond to an edge. In both these cases, the condition $s_X^u \cap s_u^X \cap \mathcal{N}_u = \emptyset$ holds. When the face routing starts this is true as the greedy region is empty, whereas on the other situation it holds due to the empty region Property 3 (see Fig. 6).

Proposition 4 *Let $u, X \in V(G)$ and $j = s_X^u$ such that $s_X^u \cap s_u^X \cap \mathcal{N}_u = \emptyset$, left returns true if uv is on the left of uX as below;*

$$\textbf{\textit{left}}(u, X, v) = \begin{cases} \text{TRUE} & if\,(X <_2 v \,\&\, j = 1) \\ \text{TRUE} & if\,(X >_1 v \,\&\, j = 2) \\ \text{TRUE} & if\,(X <_3 v \,\&\, j = 3) \\ \text{TRUE} & if\,(X >_2 v \,\&\, j = 4) \\ \text{TRUE} & if\,(X <_1 v \,\&\, j = 5) \\ \text{TRUE} & if\,(X >_3 v \,\&\, j = 6) \\ \text{FALSE} & otherwise \end{cases} \tag{5}$$

Proof Consider the empty regions imposed by the properties of the planar graph as illustrated in Fig. 7. Let $j = 1$, by observation it is clear that a node v to the left of uX follows $X <_2 v$. Similarly when $j = 2$, a node v to the left of uX follows $X >_1 v$ and when $j = 3$ it follows $X <_3 v$. Sectors 4, 5, and 6 follow opposite relations.

It is important to note that, left function is similar to the right function in its definition. Both are defined based on the empty region property and uses a single

coordinate to determine the node to the left or right. It should be distinguished that right is only defined for even numbered sectors, whereas left is defined for all the sectors. Moreover, left function generalizes the computation, such that it determines if an edge is on the left of a line between two nodes (subject to the empty intersection property or an additional emptiness assumption). Nevertheless, in left function, it is possible to use right function in even numbered sectors, but for the clarity of the implementation we made them independent.

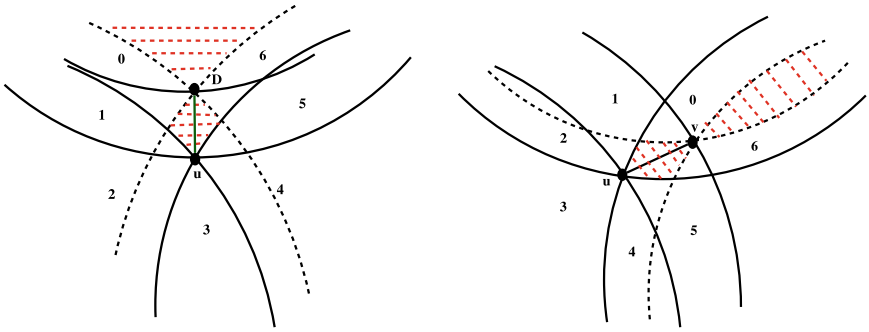
An implementation of procedure $left(u, X, v)$ is presented in Algorithm 3. By utilizing left, now we define the extended sector numbering, denoted as $num(u, X, w)$ as follows.

Algorithm 3 Implementation of the routine $left(u, X, v)$, which returns true if v is left to uX

```
1: procedure LEFT(u,X,v)
2:     j = s_u^X
3:     if j = 1 & X <_2 v then return TRUE
4:     else return FALSE
5:     end if
6:     if j = 2 & X >_1 v then return TRUE
7:     else return FALSE
8:     end if
9:     if j = 3 & X <_3 v then return TRUE
10:    else return FALSE
11:    end if
12:    if j = 4 & X >_2 v then return TRUE
13:    else return FALSE
14:    end if
15:    if j = 5 & X <_1 v then return TRUE
16:    else return FALSE
17:    end if
18:    if j = 6 & X >_3 v then return TRUE
19:    else return FALSE
20:    end if
21: end procedure
```

Definition 7 Let X be either the destination D or the node v of the incoming edge (see Fig. 7). Let k be the sector s_w^u of a given node w as defined in Definition 3 and j be the sector number of s_X^u, such that $s_X^u \cap s_u^X \cap \mathcal{N}_u = \emptyset$. Extended sector number of w is defined as;

$$\mathbf{num}(u, X, w) = \begin{cases} (k + 6 - j) \; mod \; 6 & \text{if } k \neq j \\ 0 & \text{if } k = j \;\&\; \text{left}(u, X, w) = \text{TRUE} \\ 6 & \text{if } k = j \;\&\; \text{left}(u, X, w) \neq \text{TRUE} \end{cases} \tag{6}$$



(a) When starting face routing consider the uD line

(b) When continuing face routing consider the incoming edge (u, v)

Fig. 7 Possibilities for extended sector numbering, where sector s is further divided into 0 and 6

Implementation of this function is done by first determining the number k and j according to Definition 3. Note that the nodes on the left of uX line are numbered as 0, while the node on the right of the uX line is numbered as 6 (excluding the empty regions).

Algorithm 4 Implementation of the routine $num(u, X, w)$

```
1: procedure NUM(u,X,w)
2:     k = s_u^w
3:     j = s_u^X
4:     if k ≠ j then
5:         return (k + 6 − j)  mod  6
6:     else
7:         if LEFT(u, X, w) = TRUE then
8:             return 0
9:         else
10:            return 6
11:        end if
12:    end if
13: end procedure
```

Algorithm 4 is then used to implement the next edge function, presented in Algorithm 5. In the next edge procedure, extended numbering and local ordering of nodes within a sector are used to determine the next edge. In fact, it has to compare each edge around in the neighborhood of a node in an iterative manner to determine the next edge. Algorithm 5 presents the complete procedure for determining the next edge. Lemma 1 presents the correctness of the next edge procedure. This procedure has to be incorporated into a combined greedy–face routing algorithm. Once it is possible to determine the next edge, it is possible to perform the right-/left-hand rule

to traverse around a face in the planar graph. This has to be combined with a face switching strategy to formulate the complete greedy–face routing algorithm.

Algorithm 5 Implementation of the routine *nexted ge*(u, X)

```
1: procedure NEXTEDGE(u,X)
2:     v = w ∈ N_u                                  ▷ starts with an arbitrary node
3:     for each w' in N_u \ v do
4:         if NUM(u, v)=NUM(u, w') then
5:             v=RIGHT(u, v, w')
6:         else
7:             if NUM(u, v)<NUM(u, w') then
8:                 v = w'
9:             end if
10:        end if
11:    end for
12:    return v
13: end procedure
```

Lemma 1 *Procedure nextedge in Algorithm 5 determines the edge* (u, v) *next to the line uD or incoming edge uw, provided that* $s_v^u \cap s_u^v \cap \mathcal{N}_u = \emptyset$.

Proof The proof is followed by the correctness of subroutines introduced above. According to the algorithm, it compares two edges at a time and determines which one is the next to the line uX. It keeps the next edge out of the two and iteratively compares all the other edges. Initially *nexted ge* is set arbitrarily from the neighborhood. In line 4, it checks if the two nodes w and *nexted ge* for their extended numbering based on the NUM subroutine. If the numbers are same, it uses the RIGHT subroutine to determine the rightmost edge. If the numbers are not equal, it assigns the *nexted ge* the edge with the least number. Following the proof of the $num(u, X, w)$ function, it returns the edge which is next to the uX line. Therefore, the subroutine nextedge always returns the first edge to the counterclockwise direction.

5.2.1 Face Switching

In order to formulate the complete algorithm, it is required to determine when to switch the face. According to the proof that GFG delivers data with certainty for any planar graph given in [4], when an edge (v, w) of face routing cuts the source destination line uD, face traversal must change the traversed face if the line wD is on the right of the edge (w, v). That means that the angle between (w, v) and the line wD measured counterclockwise is larger than π; i.e., we need to rotate the line (w, v) for an angle larger than π to match the line uD.

In our case, we cannot detect the intersection of the line uD. What we detect is that the edge (v, w) crosses the region $s_D^u \cap s_u^D$. By direct inspection and using the Property 2, we conclude that a crossing that triggers a face switching occurs if and

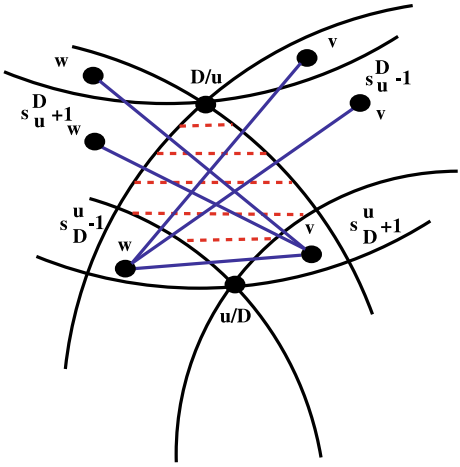


Fig. 8 There are 8 edges that can be traversed in both directions and by choosing the two possibilities for source u and destination D there are 16 different configurations leading to *face switching*. Note that only 5 edges are illustrated for the clarity

only if (the arrow $\rightarrow$ indicates the direction of the data, and the number of the sectors are all mod 6; i.e., $s_D^u + 1$ means $s_D^u \bmod 6 + 1$). Note that the sector numbering used here is the original sectoring, as we do not need to use the extended numbering to detect crossing edges.

$$\begin{aligned} &s_D^u + 1 \rightarrow s_D^u - 1, \text{ or } s_u^D + 1, \text{ or } s_u^D + 2 \\ &s_u^D - 2 \rightarrow s_D^u - 1 \\ &s_D^u + 2 \rightarrow s_u^D + 1 \\ &s_u^D - 1 \rightarrow s_u^D + 1, \text{ or } s_D^u - 1, \text{ or } s_D^u - 2 \end{aligned} \tag{7}$$

We represent on Fig. 8 the edges that can cross the region $s_D^u \cap s_u^D$. A full proof of this result proceeds by inspection. Among all the edges that cross the sector $s_D^u \cap s_u^D$, we exclude some of them by using the Property 1 illustrated in Fig. 2a. The only edges that remain and that lead to *Face Switching* following [4] are listed in Eq. (7).

The guaranteed delivery routing algorithm that we present in Algorithm 6 combines *greedy* and *face* routing in the spirit of classical greedy–face routing algorithm [4, 5] and implements the switching face algorithm from [4].

5.3 Face Routing Primitives : Geometric Approach

In this section, we present a routing algorithm [19] which guarantees delivery on the second version of the VRAC system described in Sect. 2. This approach uses geometric properties of the coordinate system to define the required geometric constructs to perform geographic routing. In fact, unlike in the previous approach, we are able

Algorithm 6 Implementation of the routine *Route(u,v,D,GREEDY)*

```
procedure ROUTE(u,v,D,mode)          ▷ u current, v previous mode = GREEDY, FACE
  if mode=GREEDY then
    next=GREEDY(u,D)
    if next = NULL then
      return NEXTEDGE(u,v)
    else
      return next
    end if
  end if
  if mode=FACE then
    next=GREEDY(u,D)
    if next = NULL then
      return NEXTEDGE(u,v)
    else
      return next
    end if
  end if
end procedure
```

to perform rotation and detection of line segment intersection as in Euclidean space. Here we summarize the basics of this approach while a complete description of the algorithm can be found in [19].

As illustrated in Fig. 2b, a node divides the physical space into six sectors based on its coordinate value. Sectors are numbered according to the Fig. 2b, and two of the components of the coordinates happen to be the borders of the sector. Drawing a similar analogy with the Euclidean space, we treat those two components as axes. Note that these axes are relative to a given sector, but the geometric properties are same in all sectors due to the symmetry. By observation, we derive some geometric properties useful for face routing, which is presented as Proposition 5 summarizing the results in [19].

Consider any sector s of a given node $u = (x_u, y_u, z_u)$ and a neighboring nodes $v = (x_v, y_v, z_v)$ and $w = (x_w, y_w, z_w)$ lying in the same sector. We observe that along a line segment in a sector, at any point on the line it preserves the ratio between perpendicular distances from two sector borders. Let θ_1 and θ_2 be the rising angles from one of the borders of the sector as in Fig. 9 and $\Delta x_1 = |x_u - x_v|$, $\Delta y_1 = |y_u - y_v|$, $\Delta x_2 = |x_u - x_w|$, $\Delta y_2 = |y_u - y_w|$ are the perpendicular distances from respective borders to the nodes.

Proposition 5 *If* $\theta_1 > \theta_2$ *then* $\frac{\Delta x_1}{\Delta y_1} > \frac{\Delta x_2}{\Delta y_2}$.

Proof We know that if $\theta_1 > \theta_2$ then $\tan\theta_1 > \tan\theta_2$ when $0 < \theta_1, \theta_2 < \frac{\pi}{2}$. With basic trigonometric relationships, we can derive $\tan\theta_1 = \frac{\Delta x_1 \sin\alpha^2}{(\Delta y_1 + \Delta x_1)\cos\alpha}$ and $\tan\theta_2 = \frac{\Delta x_2 \sin\alpha^2}{(\Delta y_2 + \Delta x_2)\cos\alpha}$, where α is the angle of the sector. With a simple algebraic simplification, it follows that $\frac{\Delta x_1}{\Delta y_1} > \frac{\Delta x_2}{\Delta y_2}$.

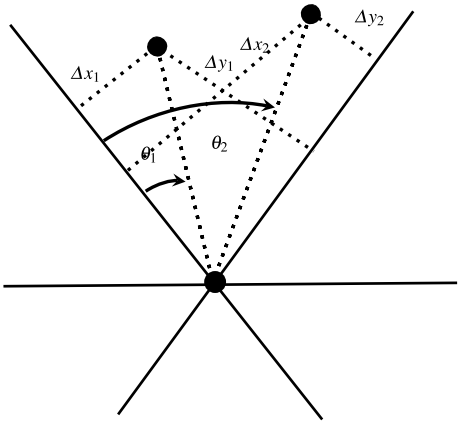


Fig. 9 Definition of the sector angle based on $\Delta x_1 = |x_u - x_v|,\ \Delta y_1 = |y_u - y_v|,\ \Delta x_2 = |x_u - x_w|,\ \Delta y_2 = |y_u - y_w|$

We use Proposition 5 to define a subroutine to decide the edge with smallest rising angle which is illustrated in pseudocode.

Algorithm 7 Comparison of angles within a sector

```
procedure RIGHT(u,v,w)                                   ▷ v, w ∈ N_u
    if Δx1/Δy1 > Δx2/Δy2 then
        return v
    else
        return w
    end if
end procedure
```

Using the subroutine 7 and sector numbers, we define the Algorithm 8 to find the first edge toward clockwise/counterclockwise direction. It starts its search from a given sector and finds the edge with smallest rising angle in that sector. If there are no edges in the sector, it continues the search through the following sectors according to their numbering. Hence, it apparently searches the whole space around a given node.

In order to perform face routing, we need to detect when a possible next edge intersects with the line segment between two other points. Due to the geometric properties of lines, it follows that two line segments get intersected, when the endpoints of the line segments do not mutually locate in the same side of the other line segment. By inspection of sector arrangement and the possibilities to satisfy this requirement, we define Algorithm 10 to check intersection of two line segments. This algorithm essentially compares the sector positions of two other points w, x compared to a selected point v with respect to point u. It repeats this process for the other combination of points and checks whether it satisfies the above-mentioned requirement.

Algorithm 8 Rotation clockwise/counterclockwise

```
procedure FACENEXTEDGE(u,D)
    i = s_D^u
    for i = 1 → 6 do
        if N_u^i ≠ ∅ then                          ▷ N_u^i = N_u in sector i
            right=RIGHT(v, w)                      ▷ v, w ∈ N_u^i
            for x ∈ N_u^i \ v, w do
                right=RIGHT(x, right)
            end for
            return right
        end if
    end for
end procedure
```

Algorithm 9 Check if two points are in the same side against a line

```
procedure ISINSAMESIDE(u,v,w,x)
    if N_u^i ≠ ∅ then
        i = S_u^w
        j = S_u^x
        if (i ≠ j) ∨ (j ≠ i + 2) then
            return (|i − j| >= 2)
        end if
    end if
end procedure
```

Algorithm 10 Check intersection based on the sector numbers and angle comparison

```
procedure ISINTERSECTING(u,v,w,x)
    if ISINSAMESIDE(u, v, w, x) then
        if ISINSAMESIDE(w, x, u, v) then
            return TRUE
        else
            return FALSE
        end if
    elsereturn FALSE
    end if
end procedure
```

Face traversal performed with the classical right-/left-hand rule, where a node rotates according to the rule and finds the first neighbor clockwise or counterclockwise. In addition to the face traversal, face changes should be performed accordingly. While there are several variants of the face changing criteria, all of them checks whether the face traversal intersects with the connecting line between current local minima and the destination. We use the defined geometric concepts to implement the face routing algorithm proposed in GFG, which is proven its delivery guarantee in an arbitrary graph. Face routing algorithm is illustrated in Algorithm 4. It uses the rotation algorithm to search its neighbors clockwise or counterclockwise and check

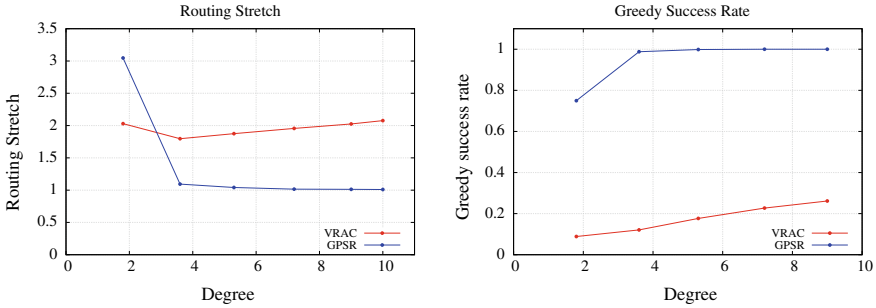
for intersections based on the Algorithm 3. If it detects an intersection, following the GFG algorithm it changes the rotation direction accordingly.

5.4 Numerical Validation

We evaluate our geographic routing algorithm in comparison with GPSR protocol. Evaluation is done in a simulation environment developed in Java, which purely focuses on routing algorithms, while ideal radio characteristics and link layer complexities are abstracted. We extract a planar subgraph of the communication graph according to the Schnyder's criteria explained earlier. For GPSR, underlying planar subgraph is a gabriel graph [11]. We analyze the *stretch factor* and *greedy success ratio*, both in hop distance and Euclidean distance by varying the average degree of a node. We simulate two settings of networks: one with a random distribution of nodes and the other with random obstacles. Nodes are distributed over a 1000×1000 square unit area, with 50 units of radio coverage.

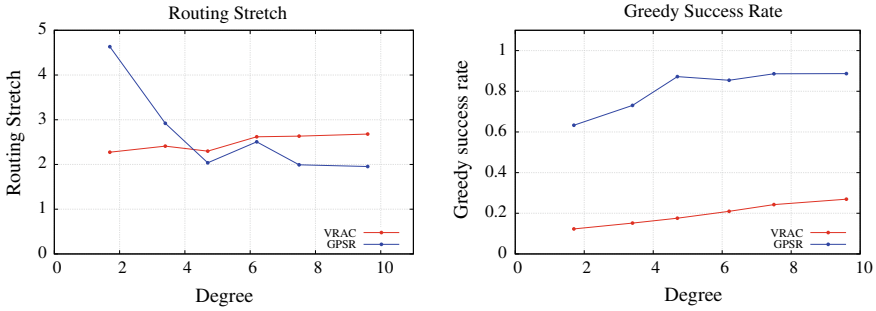
According to the Fig. 10a, VRAC performs better in stretch, when the network is sparser (degree 2). In this instance, greedy success rate of GPSR reports the lowest compared to denser instances. Thus, it has to perform face routing more often, apparently resulting in higher stretch. Comparatively, VRAC exhibits a poor greedy success (Fig. 10b) even with denser networks (roughly 8 to 22%), yet due to efficient face routing the stretch is maintained low (below 2.3 in all instances).

When the obstacles are present, VRAC reports lower stretch even for denser networks (Fig. 11a), compared to GPSR. This is due to the poor greedy success of GPSR due to the obstacles; see Fig. 11b and comparatively inefficient face routing. Overall, when obstacles are present, stretch performance is closer in both approaches.



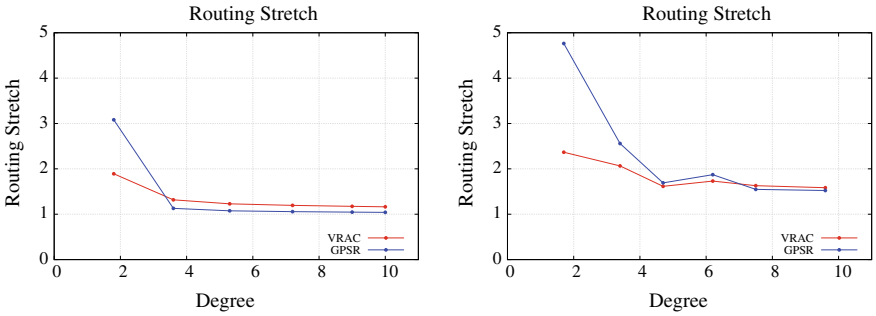
(a) Routing stretch (b) Greedy success rate

Fig. 10 Routing stretch in hop distance and greedy success rate without obstacles



(a) Routing stretch (b) Greedy Success Rate

Fig. 11 Routing stretch in hop distance and greedy success rate with obstacles



(a) Without Obstacles (b) With Obstacles

Fig. 12 Routing stretch in Euclidean distance

Finally, we analyze the stretch factor in terms of Euclidean distance; see Fig. 12a, b. More importantly, the stretch in both metrics (Euclidean and hop distance) exhibit a lower variance for VRAC across sparser and denser network instances. We emphasize that VRAC does not perform costly computations, hence reducing the localization overhead. This leads to an effective design trade-off for network design.

6 Greedy Routing over Virtual Raw Anchor Coordinates

While combined greedy–face routing guarantees delivery, it is important to investigate the conditions on VRAC, where greedy routing with delivery guarantees can be performed. In this section, we study a special case of a Schnyder graph, namely a *saturated graph*, and formulate a greedy routing algorithm.

6.1 Schnyder Characterization and Saturated Graph

Let $G = (V, E)$ represent the communication graph, where V, E are the vertex and edge sets, respectively. We investigate a network in a standard VRAC setting, hence all the properties are as described in the Sect. 2. Considering a planar subgraph extracted according to the Schnyder properties, a node u has at most one edge $(u, v) \in E$ such that $v \in s^u_{2i-1}$. Moreover, such a node v satisfies that $v <_i z \ \forall z \in s^u_{2i-1}$, i.e., $v = min_i\{z \mid z \in s^u_{2i-1}\}$. To develop the argument for a greedy algorithm, we rewrite the order relations in each sector as in Proposotion 6.

Proposition 6

$$\text{If } \mathbf{v} \in \mathbf{s^u_1},\ \mathbf{z} \neq \mathbf{v} \text{ we have } \left.\begin{array}{l} z <_2 u \\ z <_3 u \end{array}\right\} \Rightarrow z >_1 v. \tag{8}$$

$$\text{If } \mathbf{v} \in \mathbf{s^u_3},\ \mathbf{z} \neq \mathbf{v} \text{ we have } \left.\begin{array}{l} z <_1 u \\ z <_3 u \end{array}\right\} \Rightarrow z >_2 v. \tag{9}$$

$$\text{If } \mathbf{v} \in \mathbf{s^u_5},\ \mathbf{z} \neq \mathbf{v} \text{ we have } \left.\begin{array}{l} z <_1 u \\ z <_2 u \end{array}\right\} \Rightarrow z >_3 v. \tag{10}$$

As described in Sect. 4, it is possible to perform a local planarization algorithm to obtain a planar subgraph of $G(V, E)$. An immediate saturation condition on such a planar subgraph leads to the following definition.

Definition 8 Saturated graph [20] A planar graph is saturated if there exists exactly one edge in each sector s^u_{2i-1}, $i = 1, 2, 3$ for each node u.

The definition holds for planar graphs given in an abstract way. For instance, a maximal planar graph drawn as a Schnyder drawing is a saturated graph. Indeed, such a planar graph admits a Schnyder representation and the definition refers to this representation. Our greedy routing algorithm is performed on this planar saturated graph and in the rest of the text all references to the graph are implicit to this graph. We propose a metric-free characterization of a greedy path[2] and show that it guarantees delivery when the graph is saturated. We use the combinatorial properties (partial orders) to reason on the delivery guarantees of our algorithm. These combinatorial properties are derived from geometric properties of a saturated graph, yet the greedy path construction is also valid if we assume that the order relation $<_i$ are given in another (abstract) way. This is why in the rest of the paper we avoid direct reference to VRAC system and make only use of the order relations.

If the (planarized) graph is saturated, then each internal node u has exactly one edge in each sector s^u_1, s^u_3, s^u_5 and an indeterminate (0, 1, 2, ...) number of edges in the remaining sectors s^u_2, s^u_4, s^u_6. Since the representation is standard the sectors s^u_1, s^u_3 and s^u_5 contain the nodes A_1, A_2 and A_3, respectively, and are not empty. The maximality assumption implies that if there is an option of adding an edge and keeping the planarity property then the edge is present [17].

[2]Given two nodes u and v, we do not assume that we can compute the distance $d(u, v)$.

For routing from a node u to a destination $D \in s_1^u \cup s_3^u \cup s_5^u$, the natural option is to follow the edge (u, v) such that $v \in s_D^u$ (=s_1^u or s_3^u or s_5^u). Next, from v, if $D \in s_1^v \cup s_3^v \cup s_5^v$ we repeat the same strategy. However, it may happen that $D \notin s_1^v \cup s_3^v \cup s_5^v$; see Proposition 9. In this case, $D \in s_2^v \cup s_4^v \cup s_6^v$ and the existence of an edge in the sector s_D^u is not provided by the Schnyder's characterization (2). Nevertheless, in Proposition 8, we show that saturation implies the existence of an edge in the sector S_D^u.

Common approach most of the greedy routing algorithms [21, 22] follow is to compute the greedy embedding given a graph and to use the underlying metric of the respective space to perform greedy routing. However, we avoid the computation of the planar embedding and the usage of a metric function for greedy routing. For instance, [23] presents an algorithm to compute the greedy embedding of planar triangulations. We rely on the metric-free definition of greedy paths in [8] without embedding the graph. The coordinate system that we use differs from previous work [21, 24], that are based on Schnyder's characterization of planar graphs. They use Schnyder drawing as the coordinate system [17], which is more complex to compute than VRAC.

6.2 *Characterization of Greedy Paths*

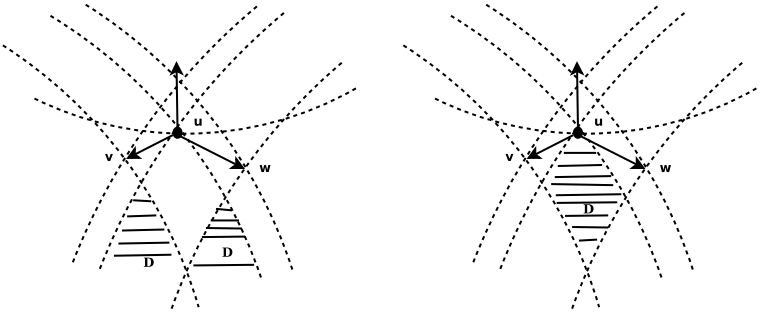
We characterize the greedy path as in Sect. 5.1, following the transitivity and odd-symmetry properties.

Definition 9 For destination node D, a path $\{u^k\}$ is a greedy path if there exists $i \in \{1, 2, 3\}$ such that

$$\forall k \;\; u^{k+1} <_i u^k, \text{ or } \forall k \;\; u^{k+1} >_i u^k. \tag{11}$$

For a greedy path, there is a coordinate that changes monotonically.

It is important to distinguish this definition of a greedy path compared to the Definition 6. This is a generalized definition as it only considers the existence of a monotonically increasing or decreasing coordinate along the path. In the following, we build greedy paths from u to D such that $u <_i u^k <_i D$ the fact that D is an upper bound and the construction continues while $u^k <_i D$ implies the convergence of the sequence to D. In the proofs of Propositions 8 and 9, we use the assumption that the graph is saturated, to say that given a node u there exist neighboring nodes in the sectors s_1^u, s_3^u, s_5^u. Unfortunately, we must proceed with caution if the node u is one of the distinguished nodes A_1, A_2, A_3 since these nodes may not have any neighboring nodes in these sectors. Actually, these nodes do not cause any trouble because there is a path from any internal nodes to them with increasing coordinate $<_i$, respectively. They are also all connected with each other. For these reasons, and in order to simplify the exposition, we no longer make any reference to these particular nodes in the proofs.



(a) Destination $D \in s^v_{2i}$ *or* s^w_{2i} (b) Destination $D \in s^u_{2i+1}$ *and* $u \in s^D_{2i-2}$

Fig. 13 Two cases to consider in greedy path construction

In the proof of Proposition 8, we use following Proposition.

Proposition 7 *If* $D' \in s^D_i$ *and* $D'' \in s^{D'}_i$ *then* $D'' \in s^D_i$

Proof This property follows directly from the transitivity of the inequalities in the definition of the sectors (3).

Proposition 8 *We assume that the graph G is saturated. Then provided that the destination D belongs to* s^u_2 *(or* s^u_4*, or* s^u_6*) then there is a path* $\{u^i\}$ *in G with* $u^0 = u$ *such that* $u^{i+1} \in s^{u^i}_2$ *(*$u^{i+1} \in s^{u^i}_4$ *or* $u^{i+1} \in s^{u^i}_6$ *respectively), and the path converges to D.*

Along the path, the order $<_3$ *(*$<_1$*,* $<_2$*) decreases monotonically if* $D \in s^u_2$ *(*$D \in s^u_4$*,* $D \in s^u_6$ *respectively).*

Proof For concreteness, we consider $D \in s^u_4$. If u is connected to D we define $u^1 = D$ and the proposition is true. Otherwise, we prove below that there exists a neighboring node of u, u^1 such that $D \in s^{u^1}_4$ and $D <_1 u^1 <_1 u$. Hence, by applying the construction iteratively, we construct the sequence of points that satisfy $u^{i+1} \in s^{u^i}_4$, lower bounded by D and decreases with respect to $<_1$, i.e., $D <_1 u^{i+1} <_1 u^i$. Such a sequence converges to D.

Let us prove that, given u such that $D \in s^u_4$, there exists x such that $(u, x) \in E$, $D \in s^x_4$ and $D <_1 x <_1 u$. u is internal, by the assumption on saturation, there exist two neighboring nodes of u such that $v \in s^u_3$ and $w \in s^u_5$. we then have (Fig. 13)

$$D <_1 u,\ D >_2 u,\ D >_3 u \ \Leftrightarrow D \in s^u_4 \tag{12}$$

$$v <_1 u,\ v >_2 u,\ v <_3 u \ \Leftrightarrow v \in s^u_3 \tag{13}$$

$$w <_1 u,\ w <_2 u,\ w >_3 u \ \Leftrightarrow w \in s^u_5 \tag{14}$$

If v (or w) is such that $D \in s^v_4$ (or $D \in s^w_4$) the next point on the path is $u^1 = v$ (or $u^1 = w$) and (13) shows that $v = u^1 <_1 u$, and $D \in s^v_4 \Rightarrow v >_1 D$ (or (14) shows that $w = u^1 <_1 u$, and $D \in s^w_4 \Rightarrow w >_1 D$).

Otherwise, we have to prove that there exists a neighboring node of u in the sector s_4^u that satisfies the conditions. We have that $D >_2 u >_2 w$, and $D >_3 u >_3 v$ [using (12–14)] and $D \notin s_4^v$ and $D \notin s_4^w$ imply

$$D \notin s_4^v \Rightarrow \left.\begin{array}{l} D >_1 v\ D <_2 v\ D >_3 v \\ D <_1 v\ D <_2 v\ D >_3 v \end{array}\right\} \text{ or } \Rightarrow D <_2 v \tag{15}$$

$$D \notin s_4^w \Rightarrow \left.\begin{array}{l} D >_1 w\ D >_2 w\ D <_3 w \\ D <_1 w\ D >_2 w\ D <_3 w \end{array}\right\} \text{ or } \Rightarrow D <_3 w \tag{16}$$

Next, because $D \in s_4^u \Rightarrow u \in s_1^D$ and the assumption of saturation, there exists an edge (D, D') with $D' \in s_1^D$. If $D' = u$, we are done.

Otherwise, by the Property (8) and $u \in s_1^D$ we have that $\boxed{D' <_1 u}$.

By gathering the inequalities corresponding to $u \in s_1^D$ with the ones deduced from (15),(16), we obtain $D <_1 D'$, $v >_2 D >_2 D'$, $w >_3 D >_3 D'$. Using $D' <_1 u$, $D' <_2 v$ with Property (9), we obtain $\boxed{D' >_3 u}$.

Last from $D' <_1 u, D' <_3 w$ and Property (10) (with edge (u, w) instead of (u, v)) we obtain $\boxed{D' >_2 u}$. Finally, we have proved that $D' \in s_4^u$ with the boxes equations and $D <_1 D' <_1 u$. The node D' plays the same role as D in the statement of the proposition but with an increasing $<_1$ order position. Because $D' <_1 u$, we find by applying iteratively the construction a sequence $D', D'', \ldots$ of nodes that converges to u and all belonging to s_4^u. Moreover, along the sequence we have $D' \in s_1^D, D'' \in s_1^{D'}$,... and Proposition 7 implies that all the points in the sequence belong to s_1^D. In particular, for the point x that is connected to u $x \in s_1^D \Leftrightarrow D \in s_4^x$. We have then proved the existence of a point $x \in s_4^u$ that satisfies $D \in s_4^x$ and such that $D <_1 x <_1 u$.

Remark 1 Construction of the greedy path if $D \in s_{2i}^u$

In order to route from u to $D \in s_{2i}^u$, the node u must first check whether for $v \in s_{2i+1}^u$ and $w \in s_{2i-1}^u$ one of the condition $D \in s_{2i}^v$ or $D \in s_{2i}^w$ is satisfied and if yes sends the message accordingly. Otherwise, the message is forwarded to (the existing) neighboring node in $x \in s_{2i}^u$ such that $D \in s_{2i}^x$. This routing scheme converges because the coordinate i decreases along the path and the path does not step over D, as all the points in the path are $>_1 D$.

Proposition 9 *Let us assume that $(u, v) \in E$ and $D, v \in s_1^u$ (or s_3^u or s_5^u). Then, $D \notin s_3^v \cup s_4^v \cup s_5^v$ (or $s_1^v \cup s_5^v \cup s_6^v$ or $s_1^v \cup s_2^v \cup s_3^v$).*

Proof Let us consider $v, D \in_1^u$ the other cases are proved similarly by a permutation of the indices. We have

$$v \in s_1^u \Leftrightarrow u <_1 v\ \ u >_2 v\ \ u >_3 v$$
$$D \in s_1^u \Leftrightarrow u <_1 D\ \ u >_2 D\ \ u >_3 D$$

Part b of the Schnyder's conditions (2) implies that D must be larger than u and v for one order and we see on the two inequalities above that it can only be $<_1$. The

condition $D \in s_3^v \cup s_4^v \cup s_5^v$ implies that $v >_1 D$ and hence there is no $i \in 1, 2, 3$ such that $u, v <_i D$ and the result is proved.

Remark 2 Construction of the greedy path if $D \in s_{2i-1}^u$
The practical implication of Proposition 9 for routing is to prove the existence of a greedy path from u to $D \in s_{2i-1}^u$. We decompose the construction in two parts and for concreteness we consider $D \in s_1^u$.
Part 1. The maximality assumption implies the existence of a node $v \in s_1^u$ such that $(u, v) \in E$. If $v = D$, we are done. Else, u sends the message to v and the first coordinate $<_1$ increases, the second one $<_2$ and the third one $<_3$ decrease. If $D \in s_1^v$ then v repeats the same procedure and the coordinates continue to be updated monotonically and $D >_1 v$ because $D \in s_1^v$. This implies that the first part of the construction converges to D or switches to the second part.
Part 2. If the path reaches a node v such that $D \notin s_1^v$ the construction of the path continues with this second part. In this case, $D \in s_2^v$ or $D \in s_6^v$ must be satisfied because of Proposition 9. In both cases, we have $D >_1 v$ and we can apply Proposition 8 that shows the existence of a sequence of nodes v' with $D \in s_2^{v'}$ or $D \in s_6^{v'}$, respectively, and this sequence eventually reaches D. If $D \in s_2^{v'}$ then by Proposition 8 the coordinate $<_3$ continues to decrease along the second part of the construction. If $D \in s_6^{v'}$ the coordinate $<_2$ continues to decrease. In both cases, we have shown that along the two parts of the construction one coordinate ($<_2$ or $<_3$) decreases monotonically and the resulting path is then *greedy*.

6.3 Routing in Maximal Planar Graph

Our results are summarized in the Theorem 2 below. The proof is apparent from the above sections. The pseudocode of the algorithm is provided in Algorithm 11, and the correctness of the algorithm is proved in the Remarks 1 and 2 of the construction of the path if $D \in s_{2i}^u$ or $D \in s_{2i+1}^u$ that follow the Propositions 8 and 9.

Theorem 2 1. *There is a greedy routing algorithm on every saturated planar graph (saturated version).*
2. *Every Schnyder drawing of a planar triangulation is a greedy embedding.*

This result is connected to previous ones. In [21], it is proven the existence of an embedding of the graph on a plane such that greedy routing is successful (using the natural metric). In [24], the authors define a routing algorithm to route messages in maximal planar graph. Their algorithm design is similar to the one we propose in Fig. 11 for maximal planar graph. It makes use of Schnyder's coordinate as defined in [17]. The computation of Schnyder's coordinate uses the *realizer* as defined in [17] and is costly. Our algorithm uses the *realizer* for routing and does not require the extra computation of the Schnyder's coordinate system. Moreover, it may be called *greedy* since we are able to introduce greediness without considering a metric. Our algorithm assumes that the underlying graph is *saturated*; see Definition 8, a

Algorithm 11 Pseudocode of the greedy routing

```
INPUT Source u, Destination D
repeat
  if D ∈ N_u then u = D                                   ▷ N_u is the set of neighbors of u
  else
    if D ∈ s^u_{2i-1} then
      u = v ∈ s^u_{2i-1} s.t. (u, v) ∈ E                  ▷ v is unique
    else                  ▷ D ∈ s^u_{2i} consider v ∈ s^u_{2i-1} and w ∈ s^u_{2i+1} s.t. (u, v), (u, w) ∈ E
      if D ∈ s^v_{2i} then
        u = v
      else
        if D ∈ s^w_{2i} then
          u = w
        else
          u = x ∈ s^u_{2i} s.t. D ∈ s^x_{2i}               ▷ must exist by Proposition 8
        end if
      end if
    end if
  end if
until u=D
```

property satisfied by Schnyder's drawing [25]. In [26, 27], the authors complement the work in [24] by computing a metric ensuring that 3−connected graphs admit a greedy embedding.

References

1. Perkins, C.E., Royer, E.M.: Ad-hoc on-demand distance vector routing. In: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications, p. 90. IEEE Computer Society (1999)
2. Haas, Z.J.: A new routing protocol for the reconfigurable wireless networks. In: 1997 IEEE 6th International Conference on Universal Personal Communications Record. Conference Record, 1997, vol. 2, pp. 562–566. IEEE (1997)
3. Bose, P., Carmi, P., Durocher, S.: Bounding the locality of distributed routing algorithms. Distrib. Comput. **26**(1), 39–58 (2013)
4. Bose, P., Morin, P., Stojmenović, I., Urrutia, J.: Routing with guaranteed delivery in ad hoc wireless networks. Wirel. Netw. **7**(6), 609–616 (2001)
5. Karp, B., Kung, H.-T.: GPSR: greedy perimeter stateless routing for wireless networks. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, pp. 243–254. ACM (2000)
6. Santoro, N., Khatib, R.: Labelling and implicit routing in networks. The Comput. J. **28**(1), 5–8 (1985)
7. Gnawali, O., Fonseca, R., Jamieson, K. Moss, D., Levis, P.: Collection tree protocol. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, pp. 1–14 (2009)
8. Li, Y., Yang, Y., Xianliang, L.: Rules of designing routing metrics for greedy, face, and combined greedy-face routing. IEEE Trans. Mob. Comput. **9**(4), 582–595 (2010)

9. Frey, H., Stojmenovic, I.: On delivery guarantees and worst-case forwarding bounds of elementary face routing components in ad hoc and sensor networks. IEEE Trans. Comput. **59**(9), 1224–1238 (2010)
10. Kuhn, F., Wattenhofer, R., Zollinger, A.: Worst-case optimal and average-case efficient geometric ad-hoc routing. In: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing, pp. 267–278. ACM (2003)
11. Gabriel, K.R., Sokal, R.R.: A new statistical approach to geographic variation analysis. Syst. Biol. **18**(3), 259–278 (1969)
12. Toussaint, G.T.: The relative neighbourhood graph of a finite planar set. Pattern Recognit. **12**(4), 261–268 (1980)
13. Leong, B., Liskov, B., Morris, R.: Geographic routing without planarization. In: NSDI, vol. 6, p. 25 (2006)
14. Fonseca, R., Ratnasamy, S., Zhao, J., Ee, C.T., Culler, D., Shenker, S., Stoica, I.: Beacon vector routing: scalable point-to-point routing in wireless sensornets. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, vol. 2, pp. 329–342. USENIX Association (2005)
15. Fang, Q., Gao, J., Guibas, L.J., De Silva,V., Zhang, L.: Glider: gradient landmark-based distributed routing for sensor networks. In: INFOCOM 2005. Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, pp. 339–350. IEEE (2005)
16. Samarasinghe, K., Leone, P.: Combinatorial approach for geographic routing with delivery guarantees. In: SENSORNETS 2014—Proceedings of the 3rd International Conference on Sensor Networks, Lisbon, Portugal, 7–9 January, 2014, pp. 195–204 (2014)
17. Schnyder, W.: Planar graphs and poset dimension. Order **5**(4), 323–343 (1989)
18. Huc, F., Jarry, A., Leone, P., Rolim, J.: Efficient graph planarization in sensor networks and local routing algorithm. In: 2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 140–149. IEEE (2012)
19. Samarasinghe, K., Leone, P.: Geographic routing with minimal local geometry. In: 2012 IEEE 18th International Conference on Parallel and Distributed Systems (ICPADS), pp. 901–906. IEEE (2012)
20. Leone, P., Samarasinghe, K.: Greedy routing on virtual raw anchor coordinate (vrac) system. In: 2016 International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 52–58. IEEE (2016)
21. Dhandapani, R.: Greedy drawings of triangulations. Discret. Comput. Geom. **43**(2), 375–392 (2010)
22. He, X., Zhang, H.: Schnyder greedy routing algorithm. In: Theory and Applications of Models of Computation, pp. 271–283. Springer (2010)
23. Angelini, P. Frati, F., Grilli, L.: An algorithm to construct greedy drawings of triangulations. J. Graph Algorithms Appl. **14**(1), 19–51 (2010)
24. He, X., Zhang, H.: A simple routing algorithm based on schnyder coordinates. Theor. Comput. Sci. **494**, 112–121 (2013)
25. Leone, P., Samarasinghe, K.: Every Schnyder Drawing is a Greedy Embedding. arXiv:1609.04173 (2016)
26. He, X., Zhang, H.: On succinct convex greedy drawing of 3-connected plane graphs. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1477–1486. SIAM (2011)
27. Wang, J.-J., He, X.: Succinct strictly convex greedy drawing of 3-connected plane graphs. Front. Algorithm. Algorithm. Asp. Inf. Manag. 13–25 (2012)

Delay-Tolerant Mobile Sensor Networks: Routing Challenges and Solutions

Eyuphan Bulut

Abstract Delay-tolerant Mobile Sensor Networks (DTMSNs), which have features of both Delay-tolerant Networks (DTNs) and Wireless Sensor Networks (WSNs), need to be considered as a different network type due to the unique characteristics. DTMSNs have been getting popular due to the increasing number of applications. As a result, several routing algorithms for the communication between the nodes have been developed recently. In this chapter, we discuss the challenges for routing in the DTMSN environment and present a survey of existing routing algorithms in the literature. We categorize the DTMSNs as terrestrial, underwater, and flying DTMSNs and go through the challenges and routing solutions in each of these subcategories. We not only examine the routing algorithms specifically designed for DTMSNs but also examine the routing algorithms designed for DTNs and WSNs from the perspective of DTMSNs. Moreover, we discuss the evaluation metrics used for the performance analysis of developed routing algorithms.

1 Introduction

Wireless Sensor Networks (WSNs) have become popular with the advances in wireless communication electronics that enabled the development of low-power and small-size sensor nodes. A WSN consists of many sensor nodes deployed in a geographical area. There is a wide range of application areas of sensor networks including military networks (e.g., battlefield surveillance), environmental monitoring (e.g., habitat exploration, pollution detection), and transportation (e.g., vehicle identification and tracking). WSNs have been broadly studied in the past two decades, with primary focus on routing, energy saving, and topology control. However, when the sensors are located at moving objects such as people, animals, and vehicles and the connectivity between the objects shows DTN characteristics, the developed

E. Bulut (✉)
Department of Computer Science, Virginia Commonwealth University,
401 West Main St., Richmond, VA 23284, USA
e-mail: ebulut@vcu.edu

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_17

approaches usually fail to perform properly, requiring the development of new routing algorithms.

It has been more than a decade since Kevin Fall gave a talk about DTNs [1] and initiated the research efforts on the topic. DTN topologies are very sparse, and the nodes are connected intermittently. Thus, the network suffers from frequent partitions and the probability that there will be an end-to-end path from a source node to a destination node is very low. DTNs are originally proposed for communication for the interplanetary Internet; however, it has been applied in many different challenging environments with similar characteristics such as ad hoc and sensor networks, and vehicular networks.

Recently, DTN concept and technology has been introduced to sensor networks to address the challenges in data gathering and dissemination in mobile sensor networks with occasional connectivity. Delay-tolerant Mobile Sensor Networks (DTMSNs) is a new type of sensor network based on DTN communication principles. DTMSNs have been recently studied due to the lack of appropriate mechanisms that can handle their unique characteristics. In a DTMSN, there could be many sensors attached to the mobile devices. For example, smartphones carried by people are a good example of nodes in a DTMSN with multiple sensors (e.g., accelerometer, barometer, camera). Moreover, a group of people in a community with wearable sensor units on their body also forms a DTMSN. The connectivity between nodes can happen only when the devices are within their communication (e.g., Wi-fi, Bluetooth) range of each other. Thus, a loosely connected mobile sensor network topology is generated. Each sensor on the device generates different types of data and can be delivered to a sink node which can be located at a popular location to increase the likelihood of meeting with other nodes in the network [2]. Such sink nodes are also usually assumed to have no power problem compared to the regular nodes. Therefore, they can perform heavy processing on the collected data (e.g., filtering, aggregation). Moreover, they are usually equipped with powerful communication hardware.

In Fig. 1, a typical architecture for a DTMSN is illustrated. There can be nodes with different mobility behaviors. Some sensor nodes can be static and can only transmit data to mobile nodes that come to the range of them. Within mobile sensor nodes, some of them such as smartphones can be connected to the backbone network (e.g., cellular network) and communicate with each other. On the other hand, some of the mobile nodes such as wearable sensing units on human beings or animals [3] can only communicate to other mobile nodes and static nodes. The links between sensor nodes can also show constant and intermittent connectivity patterns. Most of the time, sensor nodes can only communicate with each other when they come to the range of each other. It is possible that some nodes (e.g., smartphones) can communicate with each other through backbone network. These nodes can also be referred to as high-end sink nodes [4]. Such nodes usually have sufficient power in their batteries so that they can collect, store, and process the data from other sensor nodes. They also use their power for communication with other nodes through the backbone network with their powerful wireless transceivers they have.

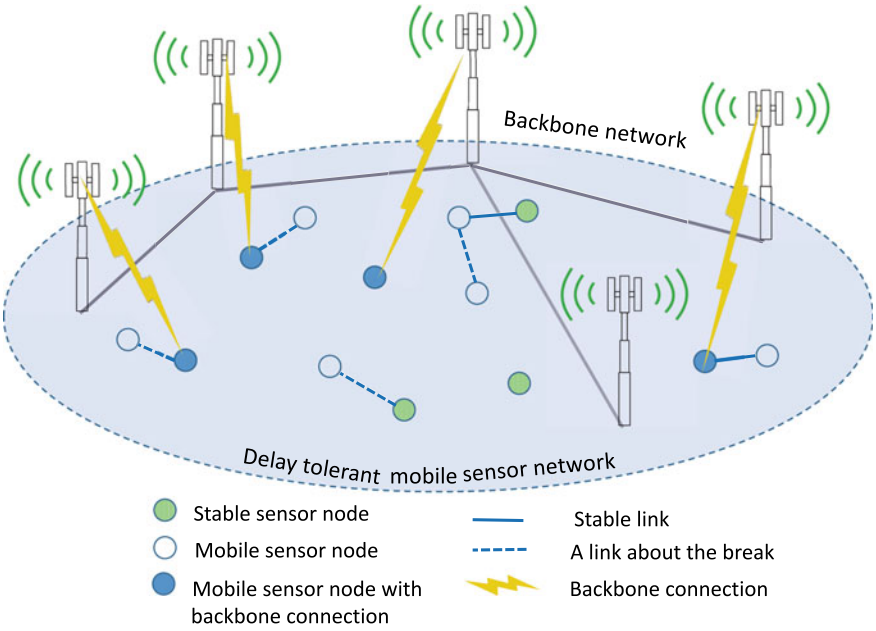


Fig. 1 Delay-tolerant Mobile Sensor Network architecture with different types of sensor nodes and varying links

DTMSNs show similar properties as sensor networks such as short range of communication, limited computation capability, and battery capacity. However, they also have the following features which are different than traditional sensor networks [5]:

- Mobility: The sensors and sinks are carried by the people who have different mobility patterns. This generates a very dynamic network topology and many partitions. Network structure is highly affected by the mobility yielding unstable link quality between nodes. On the other hand, mobility brings opportunities for reaching out other nodes in the network.
- Intermittent connectivity: The connectivity among nodes in a DTMSN is very low; thus, a very sparse topology is created. Sensor nodes can only communicate to other nodes when they are in their communication ranges mutually, which occurs very limitedly.
- Delay-tolerant: Loose connectivity between nodes causes high delays in data delivery. However, depending on the requirements of the applications, this can be usually tolerable. Ubiquitous and pervasive data gathering could be exchanged in the expense of delays.
- Fault-tolerant: In order to increase the performance of communication, there can be multiple redundant copies of the same packet in the network. Such a policy during data acquisition and routing toward the sink makes the network more robust and

fault-tolerant. Some of the packets can be lost or damaged, but the performance of data gathering and dissemination process does not worsen.

- Limited buffer: There can be multiple sensors on each node in a DTMSN. Similar to the sensor networks, each sensor has usually limited buffer or some sensors can share some limited memory. As the sensed data needs to be communicated to some high-end sink node without being deleted due to buffer problem, queue management is important and challenging in DTMSNs.

Even though DTMSNs show similar characteristics with other networking types, they differ from them in multiple aspects. Table 1 shows the comparison of a DTMSN with other networking types in terms of several features.

The primary focus of researchers studying on DTMSNs has been the routing problem. Due to the aforementioned characteristics, designing an effective routing algorithm and a data delivery scheme for DTMSNs is challenging. Many studies have been performed on how to handle the sporadic connectivity between the nodes of a DTN and provide a successful and efficient delivery of messages to the destination.

In this chapter, we study the routing challenges and solutions in DTMSNs. There are also other challenges such as developing efficient MAC protocols, queue management, and scheduling schemes, which could be found in several surveys [2, 6, 7]. We categorize the routing algorithms developed for DTMSNs based on the deployment space of the network. In some studies [4], DTMSNs are categorized as networks with static sensors, networks with managed mobile nodes, and networks with mobile sensors. However, we think that all these sensor types could be present in the same DTMSN at the same time. Thus, we categorize the DTMSNs based on the space they are deployed. In other words, we study terrestrial, underwater, and flying DTMSNs with their own routing challenges and solutions.

The rest of the chapter is organized as follows. We discuss the routing challenges and solutions in Terrestrial Delay-tolerant Mobile Sensor Networks (T-DTMSNs) in Sect. 2. Then, we look at the different challenges and solutions in Underwater Delay-tolerant Mobile Sensor Networks (U-DTMSNs) in Sect. 3. As the third category, in Sect. 4, we discuss the differences in challenges and approaches proposed in Flying Delay-tolerant Mobile Sensor Networks (F-DTMSNs). Then, the evaluation metrics

Table 1 Comparison of a DTMSN with other network types

	Topology	Mobility	Connectivity	Density	Delay tolerability
DTMSN	Very dynamic	Various speeds	Intermittently connected	Very sparse	High
WSN	Stable	Static	Mostly connected	Dense	Low
MANET	Slow	Low speeds	Mostly connected	Moderate dense	Low
VANET	Dynamic	High speeds	Mostly connected	High	Moderate

used commonly for the performance analysis of routing algorithms in DTMSNs are presented in Sect. 5. Finally, we discuss the open research issues and provide a conclusion of the study in Sect. 6.

2 General (Terrestrial) Delay-Tolerant Mobile Sensor Networks

In this section, we overview the routing algorithms proposed for DTMSNs. Most of the routing algorithms proposed generally for DTNs also apply to DTMSNs. Thus, we will study a mixed set of routing algorithms that could be applied in DTMSNs. In a DTMSN, at any given time instance, due to the high dynamic and sparse topology, the probability that there is an end-to-end path from a source to destination is low. Most of the nodes in a DTMSN are mobile and the connectivity between nodes is constructed only when the nodes come to the transmission range of each other. In a DTMSN, even though the connectivity of nodes is not constantly maintained, it can be still desirable to send a packet from a node to another node in the network. This requires the development of a routing protocol which aims to route the packets to destination node throughout the times the connections between the nodes are available. However, this cannot be achieved by traditional routing algorithms which assume the network is connected most of the time.

From routing perspective, this requires the usage of store-carry-and-forward paradigm. That is, to deliver a message to a destination (e.g., sink) node, the messages are stored at some nodes and carried until a node with better delivery probability is encountered. The message is then forwarded to the encountered node. Such a mechanism is repeated at each hop until a node with a message copy meets the destination node.

In a standard network, since the nodes are connected most of the time, the routing protocol forwards the packets in a simple way. The cost of links between nodes is mostly known or easily estimated so that the routing protocol computes the best path to the destination in terms of cost and tries to send the packets over this path. Furthermore, the packet is only sent to a single node because the reliability of paths is assumed relatively high and mostly the packets are successfully delivered. However, in DTN-like networks, routing becomes challenging because the nodes are mobile and connectivity is rarely maintained. The transient network connectivity needs to be of primary concern in the design of routing algorithms for DTNs. Therefore, routing of the packets is based on store-carry-and-forward paradigm. That is, when a node receives a message but if there is no path to the destination or even a connection to any other node, the message should be buffered in this current node and the upcoming opportunities to meet other nodes should be waited. Moreover, even a node meets with another node, it should carefully decide on whether to forward its message to that node. It is obvious that to forward a message to multiple nodes increases the delivery probability of a message. However, this may not be the right choice because it can cause a huge messaging overhead in the network which then causes

redundant energy and resource consumption. On the other hand, sending a copy of the message to a few number of nodes uses the network resources efficiently but the message delivery probability becomes lower and the delivery delay gets longer. Consequently, it is clearly seen that there is a trade-off between the message delivery ratio and the energy consumption and delivery delay in the network.

Hence, in the design of an efficient routing algorithm for DTMSNs, the following parameters need to be carefully considered: (i) the number of copies of each message that will be distributed to the network and (ii) the determination of next hop nodes to which the message is either replicated or forwarded.

Consider the sample delay-tolerant network illustrated in Fig. 2. The figure presents different snapshots of the network topology showing connectivity between nodes at four different times. Assume node A has a message destined to node G. Looking at the snapshots, we can easily observe that delivery of the message could be achieved by node B at time t_4 if node A forwards the message to node B at time

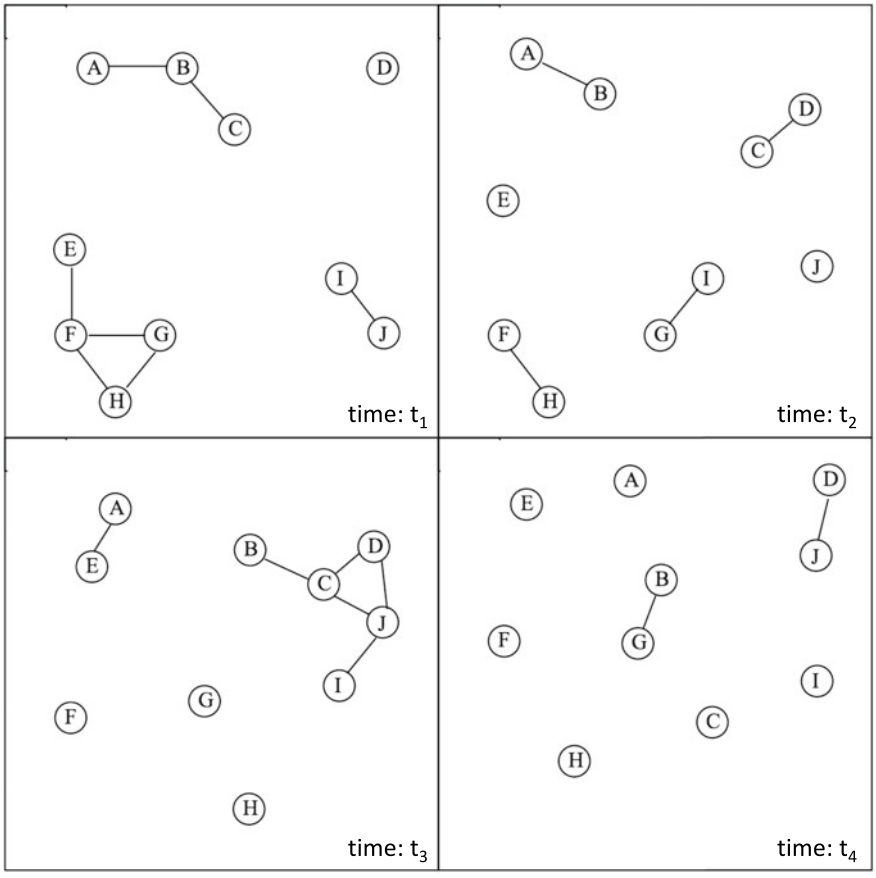


Fig. 2 Snapshots of the topology in a Delay-tolerant Mobile Sensor Network at four different times

t_1. However, the key point here is how node A will know that node B will meet the destination node before it meets the destination. What makes routing challenging in a DTMSN environment is to be able to make better decisions at contact times of nodes using only local information available at nodes.

Routing algorithms for T-DTMSNs could be classified based on the number of carriers of the message during routing. In some algorithms (i.e., single copy), only one node carries the message at all times. In these algorithms, the messages are forwarded to other nodes which are estimated to have higher chance to meet the destination. One other common method used in the design of routing algorithms for T-DTMSNs is using multiple carriers of the message. A number of copies of the same message are generated and distributed to multiple nodes so that the delivery probability of the message is increased. Among these algorithms, while some of them distribute limited number of copies [8–11] to other nodes in the network, some others [12] provide flooding-like dissemination of the message copies. Different than replication-based algorithms, some algorithms [13, 14] use erasure coding technique for efficient routing of messages. They first process and convert a message of k data blocks into a large set of blocks such that the original message can be constructed from a subset those blocks. Then, each of these encoded blocks is distributed to the other nodes in the network and the delivery of sufficient number of blocks is expected to reconstruct the original message. Finally, some routing algorithms that are designed based on social network features of these networks can also be considered as a different category. Table 2 summarizes these categories and highlights their characteristics and comparison. Next, we will discuss the details of some of the state-of-the-art algorithms in each category.

2.1 Replication-Based Routing

The very initial algorithm in this category is epidemic routing [12]. This field has attracted considerable attention after this study, and other routing algorithms are

Table 2 Routing algorithm categories for T-DTMSNs

	Number of carriers	Distribution technique	Pros	Cons
Replication-based	Multiple	Copying to met node	Fast delivery	Could be costly
Utility-based	Single	Forwarding to met node	Cost-efficient	Could be slow
Erasure-coding-based	Multiple	Distributing encoded blocks to met node	More reliable	Computation cost
Social-based	Multiple/single	Forwarding/copying to met node	Network structure-aware	Human-based networks only

developed to mitigate the problems of epidemic routing. Epidemic Routing protocol works based on principles of spreading of an epidemic. That is, when the nodes in the network come to the range of each other, nodes exchange pairwise information (i.e., their ids, ids of messages they hold) and decide which messages to share/exchange to one another. For example, at the meeting of two nodes, a summary vector that holds the index of all messages in the first node is transferred to the second node. The second node then checks its own message ids and detects the messages which are not available in its own buffer and requests the transfer of these messages from the first node. Once the messages are exchanged, two nodes have the same messages (if their contact duration allows to do so). Following this approach at every pairwise node meeting, one of the copies of the message eventually reaches the destination. As the result, the fastest spread of copies is achieved, yielding the shortest delivery time and the minimum delay.

The major drawback of this approach is excessive usage of bandwidth, buffer space, and energy due to the greedy spreading of copies. Therefore, several algorithms were proposed to limit the distribution of the message copies while still achieving high delivery rates. One of the first examples of controlled flooding-based routing algorithms is Spray-and-Wait [8] algorithm. The idea is to distribute only a limited number of copies of the same message to other nodes in the network and wait for the delivery of one of the copies of the message at the destination. The number of copies of message can be determined based on the tolerance of the application to the delay. A similar algorithm based on controlled flooding is also presented in [15]. These algorithms cannot achieve the same delivery rate as epidemic routing; however, they can achieve high delivery rates within the application's expectation, while keeping the cost of routing at very low levels. In [10, 11], Bulut et. al present the spraying-based algorithm with multiple periods. That is, the copies that will be sprayed to the network are not given to relay nodes at the beginning. First, a portion of copies are sprayed and delivery with them is waited for some time. If the delivery does not occur, in the second period, more copies are sprayed to the network. The goal is to reduce average copy count sprayed to the network while still achieving a delivery ratio by a delivery deadline. The idea is illustrated in Fig. 3.

In replication-based routing mainly unicasting is used. In sparse networks like DTMSNs, at a meeting of nodes, there are usually two nodes. Thus, unicasting is sufficient. However, in some scenarios, there can be some group of nodes meeting together and such cases can provide the opportunity to benefit from multicasting. In [16, 17], multicasting-based routing algorithms are presented. It is shown that with sufficient knowledge of network dynamics and topology information, the routing performance can increase with respect to unicasting algorithms.

2.2 Utility or Single-Copy-Based Routing

In utility-based routing, there is usually one message copy of the message. The delivery of the message is achieved through forwarding of the message between

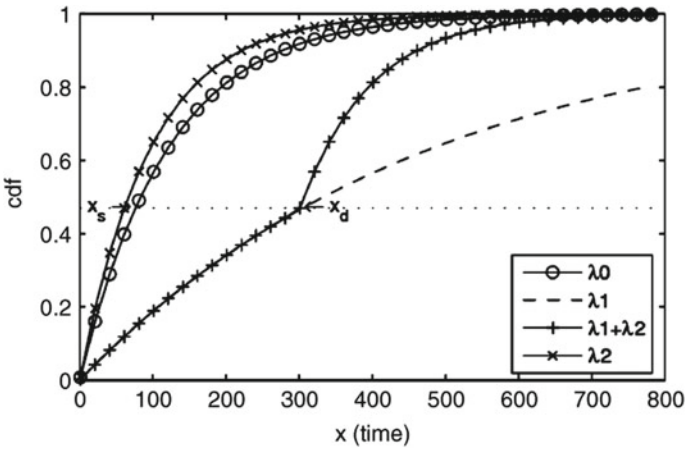


Fig. 3 cdf of delivery probability of a message when different copies are sprayed in two different periods [11]

nodes toward the destination or sink node. The utility here is the key factor that determines the forwarding of the message.

One of the first studies that addresses the weakness of epidemic routing and uses only one copy is Probabilistic ROuting Protocol using History of Encounters and Transitivity (Prophet) [18]. The protocol depends on the observation that the mobility of nodes in a DTMSN is not random and can be predictable based on repeating patterns. For example, if a node has visited a location previously several times, the probability that it will visit that location again is high. A probabilistic routing model is proposed based on this predicted mobility assumption. Each node maintains a vector of delivery predictability which shows the likelihood of meeting with other nodes. The vector is calculated based on historical meetings, and transitivity and aging mechanisms. At the meeting of two nodes, the messages are forwarded to nodes with high predictability. The delivery rate increases with the proposed idea compared to epidemic routing while achieving lower communication overhead. However, the overhead is still high; thus, there are many variants of Prophet algorithm [19, 20] which have been studied later, some of which include hybrid solutions with replications of the message.

Following the same forwarding idea, several algorithms, mainly differing from each other in terms of delivery probability computation, are proposed. For example, the time passed since the last encounter of nodes with the destination is utilized in some previous work [9, 21] and the messages are forwarded toward nodes with recent meetings with the destination. Moreover, in MaxProp [22] prioritization of the schedule of packets that will be transmitted to other nodes or that will be dropped from the buffer (due to overflow) is also taken into account in the routing decisions; thus, better performance results are achieved when the nodes have limited resources (e.g., buffer, bandwidth).

In some of the single-copy-based routing algorithms, the traditional shortest-path-based routing idea is also utilized within the DTMSN definition. That is, a virtual graph is constructed with links depending on the quality defined by a utility function. Then, the shortest path is determined from the source to the destination node and the message is routed over that path. Two example metrics used to define the link qualities are minimum expected delay (MED [17]) and minimum estimated expected delay (MEED [23]). These metrics compute the expected waiting time plus the transmission delay between each pair of nodes based on historical meetings. The first one uses the future contact schedule, and the second one uses only observed contact history. Here, note that, in shortest-path-based routing, forwarding decisions can be made at three different points: (i) at source, (ii) at each hop, and (iii) at each contact. As the utilization of recent information increases from the first to the last one, better forwarding decisions can be made. On the other hand, maintaining the link qualities with latest information and computing the updated shortest paths require more time. In [24], the impact of correlation between the meetings of a node with other meetings on shortest-path-based routing is studied. Depending on the condition of meeting with the node at previous hop, the meeting time with the next hop node is calculated; thus, the algorithm is called conditional shortest path routing. Even though such an extended algorithm improves the delivery ratio of shortest-path-based routing algorithms, there is lots of computation overhead and it may be hard to obtain link quality for each pair of nodes accurately when there is less training data of node meetings in the past. The correlation concept has also been used to develop efficient utility-based routing algorithms [25, 26] in different scenarios.

There are also hybrid algorithms that use both the idea of multiple copies of the same message and utility-based forwarding. Spray-and-Focus [9] algorithm is an example of hybrid routing protocols which basically follows the principles of Spray-and-Wait [8] algorithm but improves the process in wait duration by further forwarding the message copy to nodes that can meet the destination with high likelihood than the current holder. A similar hybrid approach is also presented in [27].

2.3 Erasure-Coding-Based Routing

In order to strengthen the robustness of the routing algorithm against failures and increase its reliability, coding-based routing algorithms have been developed. They spread many small-size messages and increase the message delivery probability. Erasure coding technique is one of the powerful coding techniques used for that purpose. The approach offers a procedure which first divides a message into k data blocks and then converts these k blocks into a set of ϕ encoded blocks such that the original message can be constructed from any $k + \epsilon$ subset of ϕ blocks. ϕ is usually set as a multiple of k, and $R = \phi/k$ is defined as replication factor of erasure coding. ϵ is considered as a very small value and can change in different coding algorithms.

Once the source node creates such encoded blocks, it distributes them to different nodes in the network and delivery of at least $k + \epsilon$ of them to the destination is

waited for the delivery of the original message. The benefit of erasure-coding-based routing is that it will not be affected much if one of the pieces is lost or corrupted as in non-fragmented-based routing. Moreover, the large files will be split into small blocks and the routing and delivery of them will be conducted separately and over different paths toward destination. All these features make the routing more robust and resilient, which is crucial in several application scenarios such as Virtual Reality (VR).

Let $p(t)$ denote the cdf of a single node's probability of meeting the destination at time t after it is generated at the source device, which will be determined based on the pairwise relations collected. The probability that there are already k messages gathered at the destination node at time t is:

$$P(t, \Phi) = \sum_{i=k}^{\Phi} \binom{\Phi}{i} p(t)^i (1 - p(t))^{\Phi - i}$$

Once the source device generates the data, it will first divide the file into small blocks of some fixed size, then with some replication factor, it will encode all these blocks to obtain the set of blocks to be forwarded toward destination. As the device meets other nodes, the blocks will be transmitted to other devices as contact duration permits. Figure 4 shows the summary of this coding and routing process from the data generated at the source device toward the target node.

The idea is introduced in [13], by Wang et al. with extensive discussion on its advantages over multicopy-based routing approach. In later studies, variants of the approach are also presented. In [14, 28], the optimal distribution of the encoded blocks over multiple delivery paths and multiple time frames are studied. In [29], a similar approach with a focus on non-uniform distribution is presented. Its application in a secure routing approach is also discussed in [30].

In [5], replication-based efficient data delivery scheme (RED), which is specifically designed for routing of messages in delay-/fault-tolerant mobile sensor networks, is presented. The RED scheme uses the erasure coding technique to reach the target data delivery rate with small messaging overhead. There are two phases of the scheme. In the first phase called data transmission, the decision of when and where to transmit the data messages are made depending on the delivery probabilities of nodes. In the second phase called message management, optimal parameters (i.e., number of blocks to encode and the redundancy level) of erasure coding technique are determined depending on the current delivery probability of the node. The RED scheme offers a simple joint message manipulation and queue management at intermediate nodes as the source computes the necessary parameters and intermediate nodes just use them. However, the optimal parameters of erasure coding calculated at the source node based on its own delivery probability to the sink node. This may result in inaccurate optimal results, especially when the source node is away from the sink [5]. To avoid these problems (e.g., increased complexity in message transmission and queue management) of RED scheme, authors propose a better scheme called Message Fault Tolerance-based Adaptive Data Delivery Scheme (FAD).

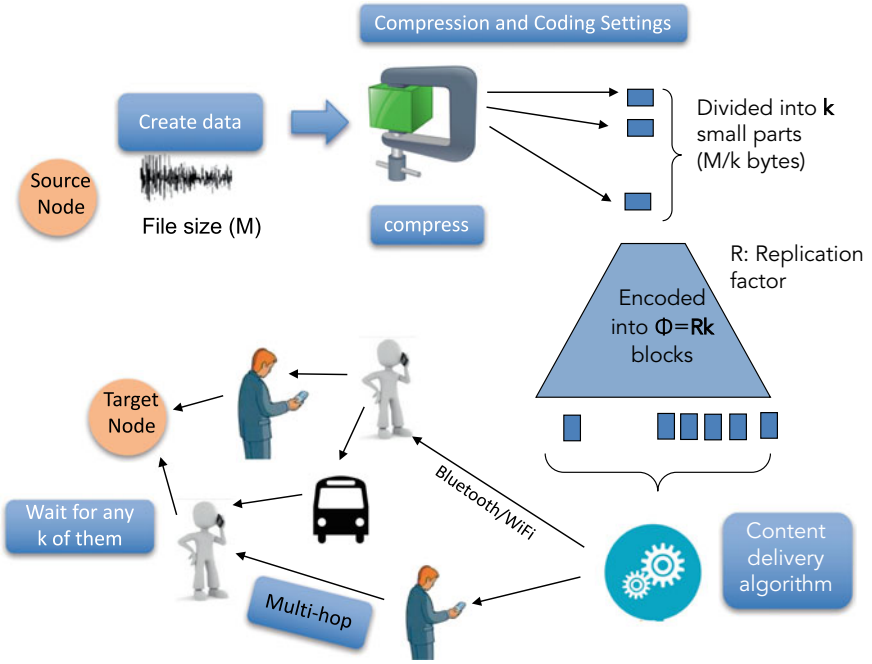


Fig. 4 Erasure-coding-based content delivery and routing from source to target node

In FAD scheme, the design is based on the message fault tolerance which is defined as the probability that at least one copy of the message is delivered to the sink by other sensor nodes in the network. Unlike the RED scheme, when a message is transmitted to next hop node, the message copy at the forwarder node is not deleted to increase the redundancy and tolerant to fault of messages. Therefore, FAD scheme can also be considered as a hybrid routing algorithm that benefits from both replication-based routing and erasure-coding-based routing. Similar hybrid approaches are also studied in several other works [27]. The message is not only encoded into small number of multiple message blocks, but these blocks are also replicated to further enhance the delivery rate.

Next, we compare the erasure-coding-based approach to replication-based (non-coding) approaches. As Fig. 5 shows, erasure coding can increase worst-case delivery ratio compared to replication-based routing (where BS stands for binary spraying and SS stands for source spraying of coded blocks). Moreover, it makes the system more reliable as the loss of a packet does not decrease the delivery ratio as it does in replication-based routing approach. Thus, each of these routing approaches can show better performance (i.e., high reliability, delivery ratio) in different environments.

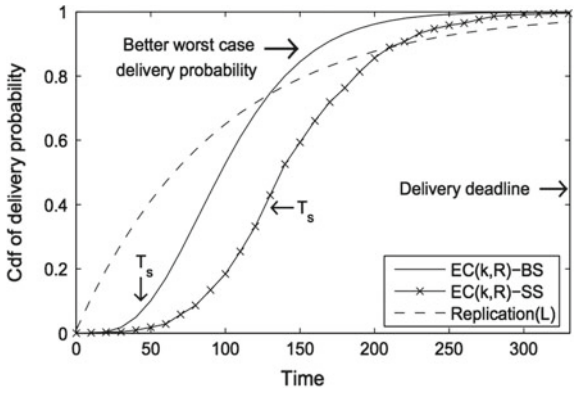


Fig. 5 cdf of delivery probability with erasure coding compared to replication (noncoding)-based approach [11]

2.4 Social-Based Routing

In some DTMSN applications, the sensor nodes are carried by human beings; thus, a human-oriented data gathering occurs. That is, the sensors on the devices attached to human body collect data related to human body, human movement, or environment in the current location of the person to achieve a goal (e.g., flu tracking, air quality detection) [32]. The data can be sensed through different devices including dedicated wearable sensing units and smartphones with multiple sensor types. With the rise of Internet of Things (IoT) and widespread adoption of smartphones, such an application forms a very large DTMSN network. Thus, designing effective routing algorithms for such networks depends on understanding human mobility and meeting characteristics (see Fig. 6 showing the highly varying node encounter patterns). To this end, several studies [25, 33–36] have been conducted on real mobile user data and identified regularities, repetitions, and correlations on these human networks.

In [37], a new data-gathering approach specifically designed for human-oriented DTMSNs is proposed. The approach estimates the delay of data transmission for each sensor node based on historical relations of humans and selectively replicates the message to sensor nodes with lower estimated delivery delay. Thus, the algorithm can be considered as a hybrid algorithm that uses replications and social relation analysis.

There are also some routing algorithms that aim to improve the routing performance based on social network properties (e.g., betweenness, centrality, degree) of the network topology. In [38], the authors use two different social network metrics to increase routing performance. First, the social similarity metric is used to detect nodes in the same community. Second, they use egocentric betweenness metric to identify the nodes that stay in between different communities and take bridging role. Then, to route a packet toward the destination or sink node, when two nodes meet each other, first a joint utility function comprised of these two metrics is calculated for each destination node. Then, the node with higher value for the message's destination is given the message.

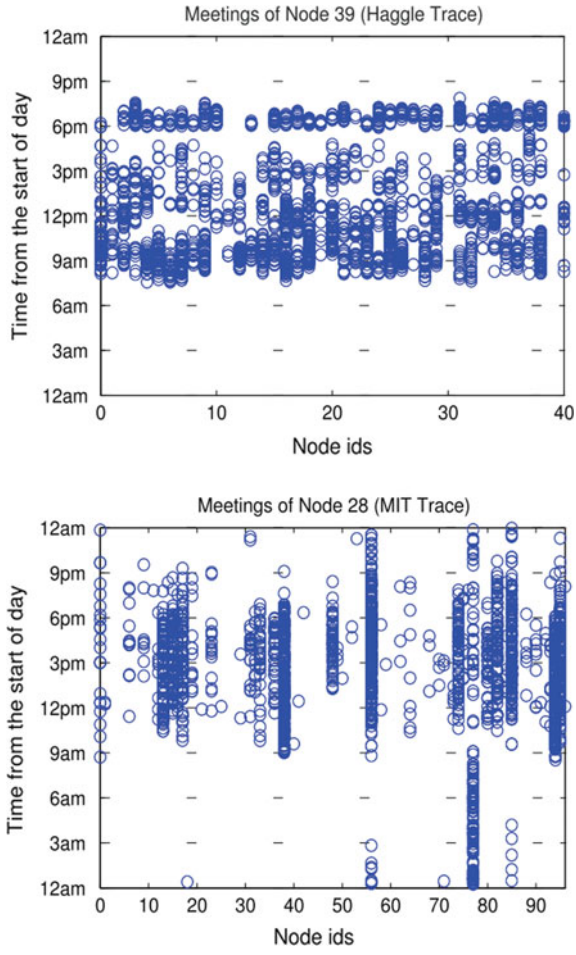


Fig. 6 Encounter distributions of selected nodes with other nodes in two different mobile social network dataset [31]

In [39], each node is assumed to have two rankings: global and local. While the former denotes the popularity (i.e., connectivity) of the node in the entire society, the latter denotes its popularity within its own community. Messages are forwarded to nodes having higher global ranking until a node in the destinations community is found. Then, the messages are forwarded to nodes having higher local ranking within destination's community. A distinction between local community members and others is also made in [40], and the distribution of message copies is optimally balanced between these two kinds of encountered nodes. In [41], a community-based epidemic forwarding scheme is introduced. First, the community structure of the network is detected using local information of nodes. Then, the message is forwarded to each community through gateways. An interesting algorithm based on the analysis of friendship relations between nodes is also proposed in [31]. Additionally, in some other studies, several different properties of social networks are considered. In [42],

irregular deviations from the habitual activities of nodes are considered and it is shown that the worst-case performance of routing can be improved by scattering multiple copies of a message in the network such that even deviant (less frequently encountered) nodes will be close to at least one of these copies. In [43], the effect of socially selfish behavior of nodes on routing is studied.

3 Underwater Delay-tolerant Mobile Sensor Networks

Recently, the area of Underwater Wireless Sensor Networks (UWSNs) has attracted a lot of attention due to high demand and rapid advances in technology. For example, mines in terrestrial areas are getting exhausted and mines in undersea terrains, which could be easier to reach than those in earth, are explored. Therefore, to detect such underwater terrain, sensors are distributed around the field of interest. Designing efficient routing protocols in UWSNs is challenging due to unique characteristics [44–46]: (i) They rely on acoustic communication (rather than RF) in which the channels offer low bandwidth and long propagation delays [47], (ii) the network topology is very dynamic as the nodes move with water currents and some nodes can be underwater autonomous vehicles, (iii) localization in underwater is difficult [48, 49]; thus, routing algorithms that need location information may not perform well, and (iv) energy efficiency in the design of routing algorithm should be a priority as the underwater sensor nodes are usually battery powered, and could be hard to recharge or replace them in such challenging environment.

Moreover, even though UWSNs have similar basic functions of sensor networks including sensing, measuring, and information collection, they may also show characteristics of delay-/disruption-tolerant networks (DTNs). In Fig. 7, a sample Underwater Delay-tolerant Mobile Sensor Network (U-DTMSN) is illustrated. The sink node usually floats on the water, while the other sensors are located at different depths of the water. There are also some mobile (e.g., underwater robot) nodes that move around and collect data from other nodes. The links between nodes can be stable and intermittent; thus, some delay is tolerable in the communication between nodes. Once the sink node receives the data from other nodes, it can connect to satellite or onshore station via RF link.

Due to the movement of sensors (e.g., triggered by flows in water), the network topology could vary a lot and can be hard to control. Such uncertainty in the network structure makes the routing of information from sensors toward the sink a challenging task. The routing protocols designed for them need to consider additional constraints compared to the routing algorithms in other DTN types. There is a significant amount of literature on routing protocols for U-DTMSNs. In Table 3, we provide a categorization of them, highlight their characteristics, and compare. Next, we will discuss a few routing protocols in each category.

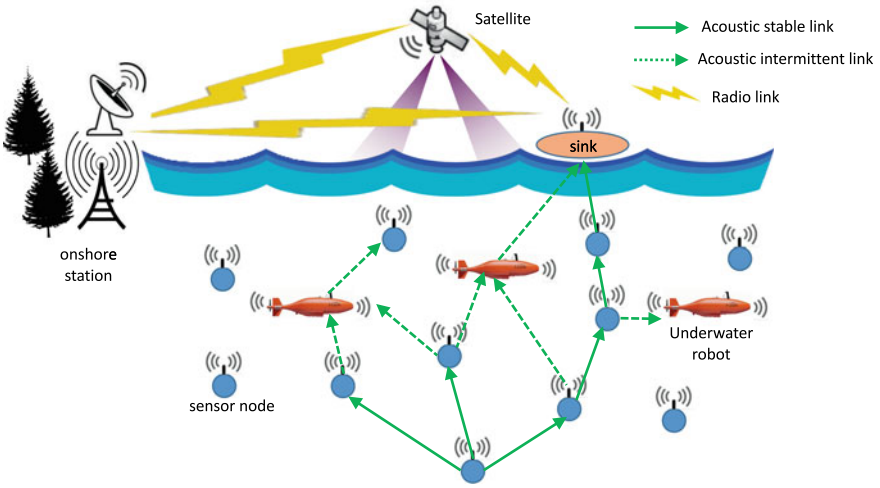


Fig. 7 Underwater Delay-tolerant Mobile Sensor Network ecosystem

Table 3 Routing algorithm categories for U-DTMSNs

	Key feature	Pros	Cons
Geographical	Uses locations of sensors	Efficient in dense networks	May not perform well with sparsity
Mobile relays-based	Uses autonomous vehicles	Fast delivery	Costly
Clustering-based	Cluster heads	Structural	Limited application
Opportunistic and prediction-based	Learning-based	Works well with sparsity	Needs warm-up time and training

3.1 Geographical Routing

The algorithms in this category utilize location information of the sensors for efficient design. In [50], Depth-based Routing (DBR) is proposed. It only utilizes the two-dimensional depth information of sensors. Multiple sink nodes are also considered on the water surface to increase the data collection process. The depth information is exploited to decide the next sensor node (i.e., qualified forwarder) to forward the packets. If the difference between the depth of the nodes is larger than a threshold, the node close to surface is considered as qualified forwarder. Even though such distance-based approach helps limit the number of forwardings of the packets and reduce the energy consumption, it takes into account only one parameter which leads some drawbacks. For example, with a large depth threshold, delivery ratio can decrease as there will be limited number of forwarders. That is why DBR algorithm will work properly only in dense networks, which may not be the case especially in UWSNs. Pressure routing protocol proposed in [51] is similar to DBR, but it uses the

pressure levels at the seafloor as the key factor rather than depth. There are also other geographical routing algorithms [52–54] which use location information of nodes. However, they have either high communication cost or cannot perform well in sparse networks.

3.2 Mobile Relays (AUVs)-Based Routing

Another type of algorithm gets benefit from mobile relays in their designs. For example, Delay-tolerant Data Dolphin (DDD) algorithm that is proposed in [55] assumes that in addition to many static sensor nodes in the seabed there are also some mobile sensor nodes (or Autonomous Underwater Vehicles (AUVs) [56]) called dolphins. The static sensors collect information from seabed and forward them to the closest dolphin node around. Dolphin nodes move randomly in the area and send beacons to static sensors periodically to notify them about their presence. Receiving the packets from static nodes, dolphin nodes deliver them to the base station once they come to the range of it. Clearly, the performance of this type of routing will depend on the number of dolphin nodes and their characteristics (e.g., speed, range). With smaller number of dolphin nodes, there will be delay in packet delivery to sink node (i.e., base station). On the other hand, with more dolphin nodes, the cost will increase. Thus, such a trade-off requires careful determination of the number of dolphin nodes based on available resources and performance requirements. There are also other studies [57] which use AUVs to aid the routing.

In [46], a Link-state-based Adaptive Feedback Routing (LAFR) algorithm is introduced using the 3D directions of underwater topology. The study analyzes the routing in underwater networks with symmetric and asymmetric links (which is determined by looking at the presence of the same node in downstream and upstream node tables) and proposes an energy-efficient routing mechanism considering this separation. Impact of multiple factors (e.g., angle, radius, interference, beam width) on link states is considered. Once the link states are determined, the routing query proceeds according to the routing information at each node. There is also a feedback mechanism, which is used to update the routing information and increase the efficiency.

3.3 Clustering-Based Routing

Classical cluster-based routing algorithms are also proposed for U-DTMSNs. In Minimum-cost Clustering Protocol (MCCP) [58], clusters are formed based on total energy required to send data to cluster heads, residual energy at clusters, and the distance between the cluster head and the sink. Even though MCCP can improve energy efficiency and prolong the network life, it does not support routing over multiple hops. Location-based Clustering Algorithm for Data (LCAD) [59] gathering

is another clustering-based algorithm which divides the entire network into 3D grids and determines the cluster head accordingly. Thus, the performance of LCAD heavily depends on the position of cluster heads, but with optimal locations, it can reduce energy consumption during data transmission phase.

3.4 Opportunistic and Prediction-Based Routing

There are also several algorithms which focus on opportunistic nature of Delay-tolerant Mobile Sensor Networks. The communication between nodes is predicted based on several parameters. Prediction-based Delay-tolerant Protocol (PBDTP) [60] is one of the algorithms in this category. It uses a unique prediction approach rather than relying on round-trip time (RTT), which may not be accurately estimated due to the obstructions and different propagation rates in the underwater environment. The algorithm can show tolerance to long and varying delays, and disruptions between nodes. It predicates a value for a sensor node if its data does not reach to sink node (instead of having the node retransmit the data). Sensors are formed into clusters. If the packet from a node to cluster head is lost, cluster head predicts it based on previous data values of the sensor node or its neighbor sensor nodes' data values. Once the cluster head node receives actual data from a sensor node, it replaces the predicted value and use the actual data value for accurate predictions in the future. PBDTP reduces data traffic in the network and uses the network resources efficiently. However, if the actual data arrival interval from the nodes becomes large, the accuracy of the proposed approach reduces dramatically.

In order to increase the delivery chance, multiple copies of the same message are sent toward the sink node. However, this can yield high overhead in the network. Thus, in some studies, the trade-off between high delivery rates and cost is analyzed. In [61, 62], a Redundancy-based Adaptive Routing (RBAR) protocol is proposed. The essential part of that algorithm is a sensor node is allowed to hold a packet as long as possible until it is necessary to make a copy. By this way, the algorithm aims to control the replication procedure and the copy count in the network, but in the meantime achieving a guaranteed in-time delivery and better resource consumption. Binary tree-based copy distribution scheme is utilized, and the packet replication process is modeled as a continuous Markov chain process with an absorbing state. The minimum number of copies required to guarantee a certain level of delay is also calculated. This is similar to the concept of Spray-and-Wait algorithm where only a certain number of copies of the message are delivered. Moreover, in [11], the idea is extended with multiple period concepts. The algorithm can achieve a good trade-off among delivery ratio, delay, and energy consumption.

Machine learning techniques are also utilized extensively in some algorithms. In [63], a reinforcement learning-based algorithm (i.e., Q-learning) is used to deal with uncertainty dynamics of UWSNs. The states are associated with each packet holding a specific packet, and actions are defined based on the forwarding of the packets between nodes. The reward in Q-learning model is defined based on residual energy

and density. With increasing density, the forwarding probability increases but also the energy consumption. Thus, a weighted equation is used. A Markov Decision-process (MDP)-based model is used to define the relationship between the states of Q-learning; then, an energy-efficient and adaptive routing protocol is proposed. Even though the algorithm increases performance, the learning algorithm needs some warm-up time to learn and converge; thus, there might be unsatisfactory performance in some cases.

Even though multicopy-based algorithms offer improved delivery ratios potentially, due to the inconsistent nature of UWSNs, some copies can be lost and do not provide benefit while increasing the cost. Thus, there are also studies which focus on routing with single-copy-based messages. In [64, 65], Guo et al. propose Prediction-assisted Single-copy Routing (PASR), in which single copy of the message is routed toward the sink over a more reliable route. The concept of aggressive chronological projected graph (ACPG), which basically integrates dynamic topology change in the same graph, is introduced. Once the historical information about links between nodes is collected and ACFG is formed (which could also be considered as mobility pattern of nodes), the optimal routes are determined for the single copy of the message. For example, the nodes with more neighbors are selected to be part of the routes as they can provide more stable connectivity with their neighbors. Other factors considered for optimal route selection include geographic location, contact periodicity, inter-contact time distribution, and contact probability.

4 Flying Delay-Tolerant Mobile Sensor Networks

Delay-tolerant Mobile Sensor Networks (DTMSNs) can also consist of flying agents (e.g., unmanned aerial vehicles (UAVs) or drones) in some scenarios. UAV systems can be operated remotely by human operators or can fly autonomously without carrying human personnel. Exploiting UAVs offers new flexible ways for many applications including military, search and rescue, public safety [66], transportation, wildfire management [67], wind estimation [68], disaster monitoring [69], remote sensing [70], and traffic monitoring [71]. Missions can be completed with UAVs faster and with less cost. Some example missions could be listed as monitoring of a disaster area and conducting an assessment of the destruction, video dissemination, victim localization through thermal cameras or wireless signals from phones, and providing immediate aid to stranded people in rural areas through video service.

Flying DTMSN (F-DTMSN) is a special network type that could be considered as a subtype of Mobile Ad hoc NETworks (MANETs), Wireless Sensor Networks (WSNs), Delay-tolerant Networks (DTNs), and Vehicular Adhoc NETworks (VANETs). However, they have unique features which make them different than these networks. As the communication range of UAVs is much higher than the nodes (e.g., smartphone carried by people) in terrestrial DTMSNs, the distances between nodes in an F-DTMSN are very large and the degree of nodes could be much higher than in other networking scenarios. As the drones are agile and can move fast, the

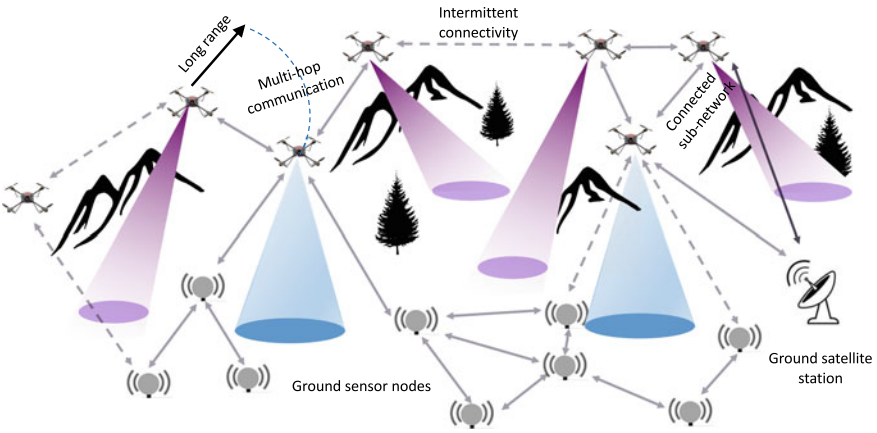


Fig. 8 Flying Delay-tolerant Mobile Sensor Network ecosystem

network topology changes very dynamically. This results in drones that connect intermittently, a characteristic of links between nodes in DTNs.

Figure 8 shows a sample Flying Delay-tolerant Mobile Sensor Network ecosystem. There are ground sensors and UAVs flying. UAVs can carry information from one subnet to another subnet with some delay. UAVs fly around obstacles; thus, the links between UAVs are mostly intermittent. There can also be a ground satellite station which communicates with flying nodes and collects data. UAVs' range is limited; thus, they can only communicate with the other nodes in their range. Moreover, the communication between UAVs can be achieved over multiple hops.

The nodes need to communicate through peer-to-peer connections (similar to MANETs) for data exchange, synchronization, and coordination with each other. They collect data from environment through their sensors and transmit to a sink node (i.e., ground station) similar to the Wireless Sensor Networks (WSNs). In a multi-UAV network, there might be many sensors located on UAVs and each sensor may have different data dissemination requirements. Moreover, the energy constraints of UAVs are much greater than the energy constraints of nodes in other nodes (e.g., a typical UAV may have battery capacity of 5200 mAh, and a flight time of about 25 min [72], while, for example, a smartphone in terrestrial DTMSN can last a day). All these features make them show some similarities with other networking types (DTN, WSN, MANET, VANET); however, the routing protocols developed for these traditional networks do not work well due to their unique aggregate characteristics.

The advantages of F-DTMSNs with multiple UAVs can be summarized as follows [73, 74]:

- Low cost: UAVs involved in such networks are usually small; thus, their acquisition and maintenance cost is low compared to other large-size flying objects [75].

Table 4 Routing algorithm categories for F-DTMSNs

	Key feature	Pros	Cons
Routing table-based	Link quality Predetermined paths	Low latency	May not be scalable
Hierarchical-based	Cluster heads	Scalable	May not perform well with high dynamicity
Geographical	Uses locations of nodes	Works well with less dynamicity	May not perform well with high dynamicity

- High-speed: UAVs can handle missions that human beings cannot handle, and as the number of UAVs in the network increases, the speed of completing the missions becomes faster [76].
- Large scalability: UAVs can connect multiple ground sensors and provide communication opportunity with minimum cost. This increases the scalability of the sensor networks and size of coverage area [77].
- High survivability: When multiple UAVs are involved in an F-DTMSN, failure of one UAV could easily be fixed through the reassignment of tasks to other UAVs and the operation survivability increases.
- Small radar cross section: Multi-UAV-based sensor networks can produce very small radar cross sections, instead of having one large radar cross section. This will be critically important especially in military-based applications [78].

On the other side, there are some challenges to design routing algorithms for these networks. The initial studies and experiments for flying networks focused on adaptions of existing routing protocols designed originally for other networks types (e.g., MANET). However, most of such algorithms do not perform well in F-DTMSN due to the unique challenges in these networks (e.g., high mobility and very dynamic link quality). Therefore, recently new algorithms that try to address these challenges are proposed specifically for these networks. Table 4 summarizes a categorization of these algorithms and their comparison. In the rest of this section, we survey over the routing algorithms in each category.

4.1 Routing Table-Based Routing

In routing table-based algorithms, there is a routing table maintained to utilize in making forwarding decisions. These algorithms could be proactive and reactive protocols and mainly depend on the rationale used in MANET routing. However, the algorithms are usually adapted to satisfy the requirements of F-DTMSNs. For example, in [79], a novel routing algorithm that promises low latency for F-DTMSN is proposed. The algorithm is the extended version of the well-known Optimized Link State Routing (OLSR) protocol [80]. Directional antennas are utilized to find the next

relay node to improve the performance of OLSR. The sensor node chooses a set of multipoint relay (MPR) nodes such that the two hop neighbors can be covered. The proposed Directional OLSR (or D-OLSR) protocol reduces the number of MPRs with directional antennas and thus reduces the messaging overhead and decreases end-to-end latency, which is an important parameter for F-DTMSNs. Another extension of OLSR protocol is proposed in [81], with a name Predictive OLSR (P-OLSR). It uses GPS information available at UAVs and link quality with ETH metric [82]. It is currently the only F-DTMSN-specific routing technique with an available Linux implementation.

There are also different variants of these algorithms proposed. In [83], a time-slotted on-demand routing protocol is proposed for F-DTMSNs. It is indeed time-slotted version of well-known Ad hoc On-demand Distance Vector Routing (AODV) algorithm [84]. In AODV, the nodes send the control packets on random access mode, while in [83] dedicated time slots are used and only one node is allowed to send data packets. The algorithm increases packet delivery ratio and reduces packet collisions; however, this happens at the expense of some decrease in usable network bandwidth.

In a UAV network, it is also important to collect data from multiple sensors on UAVs. Thus, data-centric routing algorithms can also be adopted for them (where the routing tables are static). As the UAVs could be developed specifically for some mission, the hardware (e.g., sensor) and capabilities of UAVs could be different at every mission. This makes it difficult to use a routing algorithm for any mission. Data-centric routing algorithms can be used to address this challenge, for example, through a publish–subscribe mechanism [85]. The model links the data producers (i.e., publishers) and data consumers (i.e., subscribers). The data collected from nodes toward the sink node is aggregated at intermediate nodes. Contrary to flooding-like algorithms, only the registered data types are dispatched to the data consumers. Thus, rather than point-to-point transmission, point-to-multipoint data transmission model can be utilized. Data-centric communications are usually preferred on a network with smaller number of UAVs and with predetermined path plans. Thus, a very small portion of F-DTMSN scenarios may benefit from it.

4.2 Hierarchical/Clustering-Based Routing

There are several studies that target scalable routing for any network size of F-DTMSN. They basically use hierarchical structure consisting of clusters. Each cluster has a cluster head, and all the nodes in its cluster communicate with that cluster head through direct communication link. The cluster head communicates with upper layer UAVs or satellites directly or indirectly. When a cluster head receives data from upper layers, it disseminates them to cluster members by broadcasting. Hierarchical routing algorithms provide better performance when the target area and the number of nodes in the mission are high. However, there are critical design issues such as effective cluster formation in such dynamic networks. In [86], a mobility prediction-based cluster formation algorithm is presented. As the UAVs are fast, the high mobility

of nodes requires frequent updates to clusters. In [86], authors aim to address this challenge through a mechanism that predicts the topology updates of the network. Utilizing a dictionary tree structure prediction algorithm [87] and link expiration time mobility model, they model and predict the mobility of UAVs. The cluster head is selected as the node with the highest weighted sum of all models used. The results presented show that such a cluster head selection scheme increases the robustness of the cluster and cluster heads.

Another clustering algorithm for flying UAV networks is also proposed in [88]. The clusters are first formed on the ground station using geographical information of nodes and updated during the operation of the network. Cluster heads are selected from the nodes with high degree and long connection endurance. The cluster structure is updated based on the mission information. The simulation results presented in the study show that the stability of the network increases with the proposed clustering method and routing.

4.3 Geographical Routing

If the physical position of the nodes in the network could be retrieved via GPS sensors or other type of positioning techniques, geographical routing algorithms can also be utilized for F-DTMSNs. The algorithm uses the position of the source node and the destination to decide the forwarding strategy. There are some geographical or position-based routing algorithms developed specifically for F-DTMSNs. In [89], Lin et al. propose Geographic Position Mobility-Oriented Routing (GPMOR) algorithm. In GPMOR, the node movements of UAVs are predicted with a Gauss–Markov mobility model and this information is exploited in determining the forwarding of packets. The results presented show that this algorithm can provide better latency and high packet delivery ratio compared to existing location-based routing algorithms proposed for MANETs. While some studies claim the benefit of geographical routing for F-DTMNs, some also recommend to be cautious in applying them in F-DTMSNs. In [90], authors develop a simulation framework to study geographic routing algorithms in F-DTMSNs. Their simulation results illustrate that using only greedy geographic forwarding (such as Greedy Perimeter Stateless Routing (GPSR) [91]) is not fully reliable and a combination of other methods is necessary. In [92], the power of geographical routing is combined with routing table-based (reactive) routing. A combined algorithm called Reactive-Greedy-Reactive (RGR) routing is presented to increase the performance further. The proposed RGR algorithm basically used both the location information of UAVs and reactive end-to-end paths in the decision process.

Designing efficient routing algorithms for F-DTMSNs is very challenging due to the unique features of them. Existing algorithms can only address some aspect of these networks and propose some adaptions to existing protocols to support routing in UAV networks. Thus, thorough and stable routing algorithms are still needed.

An F-DTMSN can be used to collect information from the sensors on UAVs. However, the data generation rates at these sensors together with their priority of collection at the sink node could be different. All the data collected from all these sensors needs to be routed toward the ground station (i.e., sink) over UAVs in multihop manner. Thus, collaboration between UAVs is important for coordination and collision avoidance. Designing new routing algorithms that can converge quickly in the highly dynamic F-DTMSN environment and can satisfy the needs of different data collection requirements is still an open research issue. In [74], authors discuss the potential of data-centric routing algorithms which can support multiple application scenarios simultaneously. However, it is not yet explored for F-DTMSNs.

Considering the characteristics of such networks, in the design of a routing algorithm, the following guidelines can also be considered. Routing tables can be hard to maintain in dynamic network topology environment; thus, an efficient and quickly adapting routing algorithm should not use routing tables in traditional manner. It should also need to consider the requirements of different sensors at UAVs and also intermittent connectivity patterns between UAVs. Thus, limited replication-based routing algorithms whose benefit has been shown in DTN settings can increase the performance of routing in F-DTMSNs. However, number of copies should be determined efficiently depending on the pairwise node relations [11, 31] and mechanisms that will stop copying somehow (through ACKs from destination node or with self-stopping mechanisms [93, 94]) should be developed. As UAVs can carry GPS devices, the location information of nodes is most of the time available. Thus, routing algorithm can be geographical. Finally, the remaining energy levels of the UAVs should be considered while assigning routing-related tasks to UAVs. For example, a UAV with a packet to deliver to ground station will give preference to a neighbor UAV with more energy; however, a UAV with less energy which is getting ready to visit ground station to be recharged can be preferable rather than another UAV that can provide multihop connectivity to ground station.

5 Performance Evaluation Metrics

In this part, we discuss the evaluation metrics utilized in assessing the performance of routing algorithms in Delay-tolerant Mobile Sensor Networks. At the end of this section, we also provide an overview of the performance of the algorithms presented in previous sections with respect to these metrics.

5.1 Average Delivery Ratio

Average delivery ratio is the ratio of packets received successfully at the sink node to the total number of packets originated from all source nodes. Since the network topology in a DTMSN could be very dynamic and there could be uncertainties about

the links between the nodes, it is challenging to find the next sensor node that will lead the packets to the sink. There may not be a connection between the source and sink node at all. The routing (i.e., forwarding) decisions are made through current neighbors of the packet carrying node; however, the connection between neighbor nodes may be interrupted as they move and transmission of the packets may fail. Thus, some of the packets may not arrive to the destination sink node and average delivery ratio for all messages can decrease. Average delivery ratio metric is used to ensure that the desired number of packets (i.e., information) is delivered to the sink, so that sensor network can function properly. Delivery ratio sometimes linked with some delivery deadline. Even though these networks are tolerant to delays, there is sometimes a deadline [10, 11] considered for the successful delivery of certain ratio of packets. In that case, the packets arriving to the sink node before the deadline need to achieve a certain delivery ratio required by the application.

5.2 Average End-to-End Delivery Delay

The average end-to-end delay is another significant metric used in the evaluation of routing protocols. There can be multiple services in DTMSNs, and an immediate demand can emerge for these services or applications at the sink nodes or between the sensor nodes. The communication between nodes is achieved via intermittently connected links. Thus, a delay occurs when a message from a sensor node to another node needs to be transferred. If there will be multiple hop communication, then the delays at each hop will accumulate and constitute the end-to-end delay. For all messages through the network, this will then be averaged to find the delay performance of the routing algorithm. End-to-end delay is critical metric that can also affect other factors in the network such as power consumption and buffer utilization. As the network environment is challenging due to intermittently connected links, some messages can stuck at some nodes and new messages arriving to these nodes can cause dropping of messages.

5.3 Average Delivery Cost or Messaging Overhead

Delivery cost is usually defined as the number of copies of the message distributed during routing and/or the number of forwardings of the message copy happened between nodes. Whatever the mechanism used in the routing algorithm design (e.g., single copy, multiple copy), the delivery cost covers the communication cost between the nodes. This is also directly related to the energy consumption at nodes as the transmission of a message and receiving at the other node requires energy. Thus, this metric also can be used to understand how energy efficient the routing algorithm is. Routing algorithms should also be designed such that the energy consumption due

to the routing is balanced among all nodes in the network. Fair routing [95] is one example algorithm designed primarily with this goal.

5.4 Routing Efficiency

In some studies [31], there is also a metric called routing efficiency used to evaluate the performance of routing algorithms and compare with others. Basically, this metric measures the ratio of delivery ratio to the number of copies or forwardings made throughout the routing of a packet. This gives the contribution of each copy or forwardings to the delivery ratio. For example, direct delivery method tries to achieve delivery with the copy at the source, so the cost is only one and happens at the time of delivery to destination. But the delivery ratio with direct delivery within a deadline can be very small as some node pairs may not even meet. Thus, routing efficiency of the direct delivery method will be equal to the delivery ratio, which is small. On the other hand, in epidemic routing, the delivery ratio will be very high and best possible among all algorithms. However, the cost will be very large too, as many message copies are generated and transferred over the radio between the nodes. Thus, the routing efficiency will be small too. However, there are algorithms which can provide high delivery ratios and low overheads. The routing efficiency for them will be high.

5.5 Network Lifetime

In any type of sensor networks, one of the important metrics regarding the network performance is its lifetime. Energy is consumed by different operations of the sensor network including the copying and forwarding of the messages during routing. Thus, routing algorithms should be designed in a way that it will contribute to the goal of prolonging the network lifetime. In Wireless Sensor Networks (WSNs), network lifetime sometimes considered as the time until the first node dies or the time when the network is partitioned. However, in DTMSN systems, the nodes are not always connected, and death of one node does not cause a problem in the functioning of the routing algorithm as it is designed opportunistically. Thus, the network lifetime in DTMSN environment can be defined as the time after which there will not be any communication opportunity between nodes. In other words, there can be nodes moving around but they may not come to the range of each other at all; thus, such nodes cannot send their messages to other nodes.

Comparing the performance of the algorithms presented in previous sections in terms of the presented metrics, we observe a very broad range of performance. For example, the algorithms in each category may achieve a better average delivery ratio than the algorithms in other categories depending on the application scenario and network characteristics. A general observation is that the algorithms in which the

messages are carried by many carriers provide faster delivery, yielding higher average delivery ratios and shorter end-to-end delivery delays. Moreover, hybrid algorithms that utilize multiple information about nodes can overcome the other algorithms in terms of average delivery ratio and delay. On the other hand, these algorithms may generate high delivery cost; thus, routing efficiency can reduce dramatically. If the buffer space of nodes is limited and the generated message traffic is very high, due to the drops at nodes, this may even yield lower delivery ratios and longer delays. Therefore, the algorithms that aim a delivery with single carriers of the message may overcome these multicarrier algorithms in such environments in terms of all metrics. Network lifetime, which is basically defined by other performance metrics, could also be very different for the algorithms in each category depending on the application.

6 Conclusion

In this chapter, we give a survey of Delay-tolerant Mobile Sensor Networks (DTMSNs) and the routing algorithms proposed for them. The challenges in a DTMSN, which is a sensor network with low connectivity among nodes and sparse topology, is different than the challenges in traditional WSNs. Thus, to address the requirements, several routing algorithms are proposed or traditional ones are adapted. After summarizing the characteristics of DTMSNs and their unique features and differences from other network types, we present the routing algorithms proposed for them under three categories. We classify such networks as terrestrial, underwater, and flying DTMSNs and discuss their specific constraints and challenges. Finally, we present the performance metrics used in the evaluation of routing algorithms proposed for DTMSNs.

We believe that the following issues are still open research questions and worth studying.

- A DTMSN may have different performance requirements based on the application. High delivery ratio of packets with low messaging overhead has been considered as the most significant parameter in routing performance. However, in some applications, there might be other concerns such as prioritized delivery for some significant (e.g., emergency related, keyframes in a video file) messages. There is very less focus on routing with such special requirements, and it needs further study.
- The key factor that defines the routing performance is the ability to select the next hop nodes properly. Different mechanisms that analyze and model the relations between nodes and mobility of nodes are utilized to be able to predict nodes' future encounter patterns accurately and give reasonable forwarding decisions. However, the behavior of mobile agents carrying the sensor nodes can be very different depending on the type of agent (e.g., human, animal). Thus, efficient routing algorithms in heterogeneous environments have still needed to be studied.

- Most of the research results have been obtained through specific mobility model-based or real-trace-driven simulations. There is a lack of real DTMSN test beds for real-world evaluation of proposed systems. In such a test bed, there should be heterogeneous nodes (static, mobile) and the number of nodes should be easily updatable.

References

1. Fall, K.: A delay-tolerant network architecture for challenged internets. In: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 27–34, Aug 2003. ACM
2. Wang, Y., Dang, H., Wu, H.: A survey on analytic studies of delay tolerant mobile sensor networks. Wireless Commun. Mobile Comput. **7**(10), 1197–1208 (2007)
3. Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L.S., Rubenstein, D.: Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. In: ACM Sigplan Notices, vol. 37, No. 10, pp. 96–107. ACM (2002)
4. Wang, Y., Wu, H.: Delay-tolerant mobile sensor networks. In: Encyclopedia on Ad Hoc and Ubiquitous Computing: Theory and Design of Wireless Ad Hoc, Sensor, and Mesh Networks, pp. 303–318
5. Wang, Y., Wu, H.: Delay/fault-tolerant mobile sensor network (dft-msn): a new paradigm for pervasive information gathering. IEEE Trans. Mobile Comput. **6**(9), 1021–1034 (2007)
6. Psaras, I., Wood, L., Tafazolli, R.: Delay-/Disruption-Tolerant Networking: State of the Art and Future Challenges. University of Surrey, Technical Report (2010)
7. Li, Y., Bartos, R.: A survey of protocols for intermittently connected delay-tolerant wireless sensor networks. J. Netw. Comput. Appl. **41**, 411–423 (2014)
8. Spyropoulos, T., Psounis, K., Raghavendra, C.S.: Efficient routing in intermittently connected mobile networks: the multiple-copy case. IEEE/ACM Trans. Netw. **16**(1), 77–90 (2008)
9. Spyropoulos, T., Psounis, K., Raghavendra, C.S.: Efficient routing in intermittently connected mobile networks: the single-copy case. IEEE/ACM Trans. Netw. (ToN) **16**(1), 63–76 (2008)
10. Bulut, E., Wang, Z., Szymanski, B.K.: Time dependent message spraying for routing in intermittently connected networks. In: IEEE Global Telecommunications Conference on IEEE GLOBECOM 2008–2008, pp. 1–6, Nov 2008. IEEE
11. Bulut, E., Wang, Z., Szymanski, B.K.: Cost-effective multiperiod spraying for routing in delay-tolerant networks. IEEE/ACM Trans. Netw. (ToN) **18**(5), 1530–1543 (2010)
12. Vahdat, A., Becker, D.: Epidemic routing for partially connected ad hoc networks, Technical Report CS-200006, Duke University (2000)
13. Wang, Y., Jain, S., Martonosi, M., Fall, K.: Erasure-coding based routing for opportunistic networks. In: Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking, pp. 229–236. ACM
14. Jain, S., Demmer, M., Patra, R., Fall, K.: Using redundancy to cope with failures in a delay tolerant network. In: ACM SIGCOMM Computer Communication Review, vol. 35, No. 4, pp. 109–120, Aug 2005. ACM
15. Harras, K.A., Almeroth, K.C., Belding-Royer, E.M.: Delay tolerant mobile networks (DTMNS): controlled flooding in sparse mobile networks. In: International Conference on Research in Networking, pp. 1180–1192, May 2005. Springer, Berlin, Heidelberg
16. Zhao, W., Ammar, M., Zegura, E.: Multicasting in delay tolerant networks: semantic models and routing algorithms. In: Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking pp. 268–275, Aug 2005. ACM
17. Jain, S., Fall, K., Patra, R.: Routing in a Delay Tolerant Network, vol. 34, No. 4, pp. 145–158, ACM (2004)

18. Lindgren, A., Doria, A., Scheln, O.: Probabilistic routing in intermittently connected networks. ACM SIGMOBILE Mobile Comput. Commun. Rev. **7**(3), 19–20 (2003)
19. Xue, J., Li, J., Cao, Y., Fang, J.: Advanced PROPHET routing in delay tolerant network. In: International Conference on Communication Software and Networks, 2009. ICCSN'09, pp. 411–413, Feb 2003. IEEE
20. Sok, P., Kim, K.: Distance-based PRoPHET routing protocol in disruption tolerant network. In: 2013 International Conference on ICT Convergence (ICTC), pp. 159–164. IEEE (2013)
21. Dubois-Ferriere, H., Grossglauser, M., Vetterli, M.: Age matters: efficient route discovery in mobile ad hoc networks using encounter ages. In: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 257–266, June 2003. ACM
22. Burgess, J., Gallagher, B., Jensen, D., Levine, B.N.: MaxProp: routing for vehicle-based disruption-tolerant networks. INFOCOM **6**, 1–11 (2006)
23. Jones, E.P., Li, L., Schmidtke, J.K., Ward, P.A.: Practical routing in delay-tolerant networks. IEEE Trans. Mobile Comput. **6**(8), 943–959 (2007)
24. Bulut, E., Geyik, S.C., Szymanski, B.K.: Conditional shortest path routing in delay tolerant networks. In: 2010 IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM), pp. 1–6, June 2010. IEEE
25. Bulut, E., Geyik, S., Szymanski, B.K.: Utilizing correlated node mobility for efficient routing in DTNs. In: Elsevier Pervasive and Mobile Computing (PMC), pp 150–163, Aug 2014
26. Bulut, E., Geyik, S., Szymanski, B.K.: Efficient routing in delay tolerant networks with correlated node mobility. In: Proceedings of 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), Nov 2010
27. Chen, L.J., Yu, C.H., Sun, T., Chen, Y.C., Chu, H.H.: A hybrid routing approach for opportunistic networks. In: Proceedings of the 2006 SIGCOMM Workshop on Challenged Networks, pp. 213–220, Sept 2006. ACM
28. Bulut, E., Wang, Z., Szymanski, B.K.: Cost efficient erasure coding based routing in delay tolerant networks. In: 2010 IEEE International Conference on Communications (ICC), pp. 1–5, May 2010. IEEE
29. Liao, Y., Tan, K., Zhang, Z., Gao, L.: Estimation based erasure-coding routing in delay tolerant networks. In: Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing, pp. 557–562, July 2006. ACM
30. Bulut, E., Szymanski, B.K.: Secure multi-copy routing in compromised delay tolerant networks. Springer, Wireless Pers. Commun. (2013)
31. Bulut, E., Szymanski, B.K.: Exploiting friendship relations for efficient routing in mobile social networks. IEEE Trans. Parallel Distribut. Syst. **23**(12), 2254–2265 (2012)
32. Wu, H., Wang, Y., Dang, H., Lin, F.: Analytic, simulation, and empirical evaluation of delay/fault-tolerant mobile sensor networks. IEEE Trans. Wireless Commun. **6**(9), 3287–3296 (2007)
33. Song, C., Qu, Z., Blumm, N., Barabsi, A.L.: Limits of predictability in human mobility. Science **327**(5968), 1018–1021 (2010)
34. Song, C., Koren, T., Wang, P., Barabsi, A.L.: Modelling the scaling properties of human mobility. Nat. Phys. **6**(10), 818–823 (2010)
35. Becker, R., Cceres, R., Hanson, K., Isaacman, S., Loh, J.M., Martonosi, M., Volinsky, C.: Human mobility characterization from cellular network data. Commun. ACM **56**(1), 74–82 (2013)
36. Bulut, E., Szymanski, B.K.: Understanding user behavior via mobile data analysis. In: IEEE International Conference on Communication Workshop (ICCW) 2015, pp. 1563–1568, June 2015. IEEE
37. Zhao, H., Liu, M.: A delay-based routing protocol for human-oriented delay tolerant mobile sensor network (DTMSN). In: 2012 32nd International Conference on Distributed Computing Systems Workshops, pp. 201–208, June 2012. IEEE
38. Daly, E.M., Haahr, M.: Social network analysis for routing in disconnected delay-tolerant manets. In: Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 32–40, Sept 2007. ACM

39. Hui, P., Crowcroft, J., Yoneki, E.: Bubble rap: social-based forwarding in delay-tolerant networks. IEEE Trans. Mobile Comput. **10**(11), 1576–1589 (2011)
40. Bulut, E., Wang, Z., Szymanski, B.K.: Impact of social networks on delay tolerant routing. In: Global Telecommunications Conference on GLOBECOM 2009, pp. 1–6, Nov 2009. IEEE
41. Li, F., Wu, J.: LocalCom: a community-based epidemic forwarding scheme in disruption-tolerant networks. In: 2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, pp. 1–9, June 2009. IEEE
42. Zhou, T., Choudhury, R. R., Chakrabarty, K.: Diverse routing: exploiting social behavior for routing in delay-tolerant networks. In: International Conference on Computational Science and Engineering, 2009. CSE'09. vol. 4, pp. 1115–1122, Aug 2009. IEEE
43. Li, Q., Zhu, S., Cao, G.: Routing in socially selfish delay tolerant networks. In: 2010 Proceedings of IEEE on INFOCOM, pp. 1–9, Mar 2010. IEEE
44. Akyildiz, I.F., Pompili, D., Melodia, T.: Underwater acoustic sensor networks: research challenges. Ad Hoc Netw. **3**(3), 257–279 (2005)
45. Chitre, M., Shahabudeen, S., Stojanovic, M.: Underwater acoustic communications and networking: recent advances and future challenges. Marine Technol. Soc. J. **42**(1), 103–116 (2008)
46. Zhang, S., Li, D., Chen, J.: A link-state based adaptive feedback routing for underwater acoustic sensor networks. IEEE Sensors J. **13**(11), 4402–4412 (2013)
47. Partan, J., Kurose, J., Levine, B.N.: A survey of practical issues in underwater networks. ACM SIGMOBILE Mobile Comput. Commun. Rev. **11**(4), 23–33 (2007)
48. Erol-Kantarci, M., Mouftah, H.T., Oktug, S.: Localization techniques for underwater acoustic sensor networks. IEEE Commun. Mag. **48**(12), 152–158 (2010)
49. Chandrasekhar, V., Seah, W.K., Choo, Y.S., Ee, H.V.: Localization in underwater sensor networks: survey and challenges. In: Proceedings of the 1st ACM International Workshop on Underwater Networks, pp. 33–40, Sept 2006. ACM
50. Yan, H., Shi, Z.J., Cui, J.H.: DBR: depth-based routing for underwater sensor networks. In: International Conference on Research in Networking, pp. 72–86, May 2008, Springer, Berlin, Heidelberg
51. Lee, U., Wang, P., Noh, Y., Vieira, L.F.M., Gerla, M., Cui, J.H.: Pressure routing for underwater sensor networks. In: INFOCOM, pp. 1676–1684, Mar 2010
52. Xie, P., Cui, J.H., Lao, L.: VBF: vector-based forwarding protocol for underwater sensor networks. In: International Conference on Research in Networking, pp. 1216–1221, May 2006. Springer, Berlin, Heidelberg
53. Jornet, J.M., Stojanovic, M., Zorzi, M.: Focused beam routing protocol for underwater acoustic networks. In: Proceedings of the Third ACM International Workshop on Underwater Networks, pp. 75–82, Sept 2008. ACM
54. Chirdchoo, N., Soh, W.S., Chua, K.C.: Sector-based routing with destination location prediction for underwater mobile networks. In: International Conference on Advanced Information Networking and Applications Workshops, 2009. WAINA'09, pp. 1148–1153, May 2009. IEEE
55. Magistretti, E., Kong, J., Lee, U., Gerla, M., Bellavista, P., Corradi, A.: A mobile delay-tolerant approach to long-term energy-efficient underwater sensor networking. In: 2007 IEEE Wireless Communications and Networking Conference, pp. 2866–2871, Mar 2007. IEEE
56. Seah, W.K., Tan, H.X., Liu, Z., Ang, M.H.: Multiple-UUV approach for enhancing connectivity in underwater ad-hoc sensor networks. In: Proceedings of OCEANS 2005 MTS/IEEE, pp. 2263–2268, IEEE (2005)
57. Yoon, S., Azad, A.K., Oh, H., Kim, S.: AURP: an AUV-aided underwater routing protocol for underwater acoustic sensor networks. Sensors **12**(2), 1827–1845 (2012)
58. Wang, P., Li, C., Zheng, J.: Distributed minimum-cost clustering protocol for underwater sensor networks (UWSNs). In: 2007 IEEE International Conference on Communications, pp. 3510–3515, June 2007. IEEE
59. Anupama, K.R., Sasidharan, A., Vadlamani, S.: A location-based clustering algorithm for data gathering in 3D underwater wireless sensor networks. In: 2008 International Symposium on Telecommunications, Aug 2008

60. Zhang, Z., Lin, S.L., Sung, K.T.: A prediction-based delay-tolerant protocol for underwater wireless sensor networks. In: 2010 International Conference on Wireless Communications and Signal Processing (WCSP), pp. 1–6. IEEE
61. Guo, Z., Colombi, G., Wang, B., Cui, J.H., Maggiorini, D., Rossi, G.P.: Adaptive routing in underwater delay/disruption tolerant sensor networks. In: Fifth Annual Conference on Wireless on Demand Network Systems and Services, 2008. WONS 2008. pp. 31–39, Jan 2008. IEEE
62. Guo, Z., Peng, Z., Wang, B., Cui, J.H., Wu, J.: Adaptive routing in underwater delay tolerant sensor networks. In: 2011 6th International ICST Conference on Communications and Networking in China (CHINACOM), pp. 1044–1051, Aug 2011. IEEE
63. Hu, T., Fei, Y.: An adaptive and energy-efficient routing protocol based on machine learning for underwater delay tolerant networks. In: 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 381–384, Aug 2010. IEEE
64. Guo, Z., Wang, B., Cui, J.H.: Prediction assisted single-copy routing in underwater delay tolerant networks. In: Global Telecommunications Conference (GLOBECOM 2010), pp. 1–6, Dec 2010. IEEE
65. Guo, Z., Wang, B., Cui, J.H.: Generic prediction assisted single-copy routing in underwater delay tolerant sensor networks. Ad Hoc Netw. **11**(3), 1136–1149 (2013)
66. Merwaday, A., Guvenc, I.: UAV assisted heterogeneous networks for public safety communications. In: Wireless Communications and Networking Conference Workshops (WCNCW), pp. 329–334, IEEE (2015)
67. Barrado, C., Messeguer, R., Lpez, J., Pastor, E., Santamaria, E., Royo, P.: Wildfire monitoring using a mixed air-ground mobile network. IEEE Pervas. Comput. **9**(4), 24–32 (2010)
68. Cho, A., Kim, J., Lee, S., Kee, C.: Wind estimation and airspeed calibration using a UAV with a single-antenna GPS receiver and pitot tube. IEEE Trans. Aerospace Electron. Syst. **47**(1), 109–117 (2011)
69. Maza, I., Caballero, F., Capitn, J., Martnez-de-Dios, J.R., Ollero, A.: Experimental results in multi-UAV coordination for disaster management and civil security applications. J. Intell. Robotic Syst. **61**(1–4), 563–585 (2011)
70. Xiang, H., Tian, L.: Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (UAV). Biosyst. Eng. **108**(2), 174–190 (2011)
71. Semsch, E., Jakob, M., Pavlicek, D., Pechoucek, M.: Autonomous UAV surveillance in complex urban environments. In: IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. vol. 2, pp. 82–85, Sept 2009. IET
72. Gupta, L., Jain, R., Vaszkun, G.: Survey of important issues in UAV communication networks. IEEE Commun. Surveys Tutor. **18**(2), 1123–1152 (2015)
73. Tareque, M.H., Hossain, M.S., Atiquzzaman, M.: On the routing in flying ad hoc networks. In: 2015 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 1–9, Sept 2015. IEEE
74. Bekmezci, I., Sahingoz, O.K., Temel, S.: Flying ad-hoc networks (FANETs): a survey. Ad Hoc Netw. **11**(3), 1254–1270 (2013)
75. Chao, H., Cao, Y., Chen, Y.: Autopilots for small fixed-wing unmanned air vehicles: a survey. In: 2007 International Conference on Mechatronics and Automation, pp. 3144–3149, Aug 2007. IEEE
76. Yanmaz, E., Costanzo, C., Bettstetter, C., Elmenreich, W.: A discrete stochastic process for coverage analysis of autonomous UAV networks. In: 2010 IEEE Globecom Workshops, pp. 1777–1782, Dec 2010. IEEE
77. Morse, B.S., Engh, C.H., Goodrich, M.A.: UAV video coverage quality maps and prioritized indexing for wilderness search and rescue. In: Proceedings of the 5th ACM/IEEE International Conference on Human-Robot Interaction, pp. 227–234, Mar 2010. IEEE Press
78. To, L., Bati, A., Hilliard, D.: Radar cross section measurements of small unmanned air vehicle systems in non-cooperative field environments. In: 2009 3rd European Conference on Antennas and Propagation, pp. 3637–3641, Mar 2009. IEEE

79. Alshabtat, A.I., Dong, L., Li, J., Yang, F.: Low latency routing algorithm for unmanned aerial vehicles ad-hoc networks. Int. J. Electric. Comput. Eng. **6**(1), 48–54 (2010)
80. Clausen, T., Jacquet, P.: Optimized Link State Routing Protocol (OLSR) (No. RFC 3626) (2003)
81. Rosati, S., Kruelecki, K., Heitz, G., Floreano, D., Rimoldi, B.: Dynamic routing for flying ad hoc networks. IEEE Trans. Vehicular Technol. **65**(3), 1690–1700 (2016)
82. De Couto, D.S., Aguayo, D., Bicket, J., Morris, R.: A high-throughput path metric for multi-hop wireless routing. Wireless Netw. **11**(4), 419–434 (2005)
83. Forsmann, J.H., Hiromoto, R.E., Svoboda, J.: A time-slotted on-demand routing protocol for mobile ad hoc unmanned vehicle systems. In: International Society for Optics and Photonics on Defense and Security Symposium, pp. 65611P–65611P, Apr 2007
84. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc on-Demand Distance Vector (AODV) Routing (No. RFC 3561) (2003)
85. Ko, J., Mahajan, A., Sengupta, R.: A network-centric UAV organization for search and pursuit operations. In: Aerospace Conference Proceedings, vol. 6, pp. 626–697. IEEE (2002)
86. Zang, C., Zang, S.: Mobility prediction clustering algorithm for UAV networking. In: 2011 IEEE GLOBECOM Workshops (GC Wkshps), pp. 1158–1161, Dec 2011. IEEE
87. Konstantopoulos, C., Gavalas, D., Pantziou, G.: A mobility aware technique for clustering on mobile ad-hoc networks. In: International Conference on Distributed Computing and Networking, pp. 397–408, Dec 2006. Springer, Berlin, Heidelberg
88. Liu, K., Zhang, J., Zhang, T.: The clustering algorithm of UAV networking in near-space. In: 8th International Symposium on Antennas, Propagation and EM Theory, 2008. ISAPE 2008, pp. 1550–1553, Nov 2008. IEEE
89. Lin, L., Sun, Q., Li, J., Yang, F.: A novel geographic position mobility oriented routing strategy for UAVs. J. Comput. Informat. Syst. **8**(2), 709–716 (2012)
90. Shirani, R., St-Hilaire, M. Kunz, T., Zhou, Y., Li, J., Lamont, L.: The performance of greedy geographic forwarding in unmanned aeronautical ad-hoc networks. In: Communication Networks and Services Research Conference (CNSR). 2011 Ninth Annual. IEEE (2011)
91. Karp, B., Hsiang-Tsung, K.: GPSR: greedy perimeter stateless routing for wireless networks. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking. ACM (2000)
92. Shirani, R., et al. Combined reactive-geographic routing for unmanned aeronautical ad-hoc networks. In: 2012 8th International on Wireless Communications and Mobile Computing Conference (IWCMC). IEEE (2012)
93. Chen, Z., Chao, C.: Self-stopping epidemic routing in cooperative wireless mobile sensor networks. In: 2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 1–7, IEEE (2016)
94. Dhungana, A., Bulut, E.: Timely information dissemination with distributed storage in delay tolerant mobile sensor networks, In: INFOCOM MiseNet Workshop 2017, May, 2017, Atlanta. IEEE
95. Pujol, J.M., Lopez, T., Rodriguez, P.: Fair routing in delay tolerant networks. INFOCOM 2009, IEEE (2009)

Part VII
Privacy and Security

Location Privacy in Wireless Sensor Networks

Nikolaos Baroutis and Mohamed Younis

Abstract In recent years, wireless sensor networks (WSNs) have been attracting increased attention from the research and engineering communities, motivated by applications like security surveillance, border protection, combat field reconnaissance, environment and habitat monitoring, and target tracking. In these applications, a WSN typically consists of a large population of spatially distributed sensor nodes that cooperatively monitor conditions in their surroundings and report their findings to an in situ base station (BS). The BS not only collects and analyzes the incoming data, but also interfaces WSN to a higher authority. Hence, the unique role of the BS attracts adversary's attention since it can be a single point of failure for WSN. In addition, in some applications such as target tracking, the node that detects and reports on an event could be of utmost importance for an intruder. An adversary that seeks to diminish the network utility can apply traffic analysis techniques to locate the sink of all packets (i.e., the BS) and/or the data sources in order to apply a denial of service attack, e.g., radio jamming. Countering such a threat necessitates enhanced BS and data source anonymity to conceal their identity, role, and location from external observers. This chapter analyzes the threat, highlights anonymity metrics, categorizes contemporary traffic analysis countermeasures, and discusses some of the emerging techniques.

Keywords Wireless sensor networks · Traffic analysis attack models Anonymity · Location privacy · Anti-traffic analysis techniques

N. Baroutis · M. Younis (✉)
Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, Baltimore, MD, USA
e-mail: younis@umbc.edu

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_18

1 Introduction

Networked sensors technology is one of the most important technologies for the twenty-first century [1]. Sensor nodes, networked through wireless links and deployed in large numbers, can provide unprecedented opportunities for a wide variety of military and environmental applications such as combat field reconnaissance, border protection, security surveillance, forest fire detection, and flood detection. A wireless sensor network (WSN) is usually composed of multiple sensor nodes, spread out in a large deployment area, and an in situ base station (BS). The sensor nodes monitor the environment in their vicinity by measuring ambient conditions such as temperature, lightning condition, pressure, and humidity and report their findings to the BS over wireless links. The BS collects and analyzes the sensor data and interfaces WSN to a higher authority. The employed sensor nodes are small-sized battery-operated devices. Therefore, decreasing the transmission range of a sensor node results in energy saving and consequently extends the lifetime of the network. For that reason, multi-hop routing of data packets is the most popular strategy in WSNs [2, 3]. Figure 1, shows an example of a wireless sensor network where white arrows represent the routing topology, red antennas represent the sensor nodes, and the blue computer represents the BS.

In many applications, WSN operates unattended in inhospitable areas that lack protected physical boundary; such setup makes it very difficult, if not infeasible, to

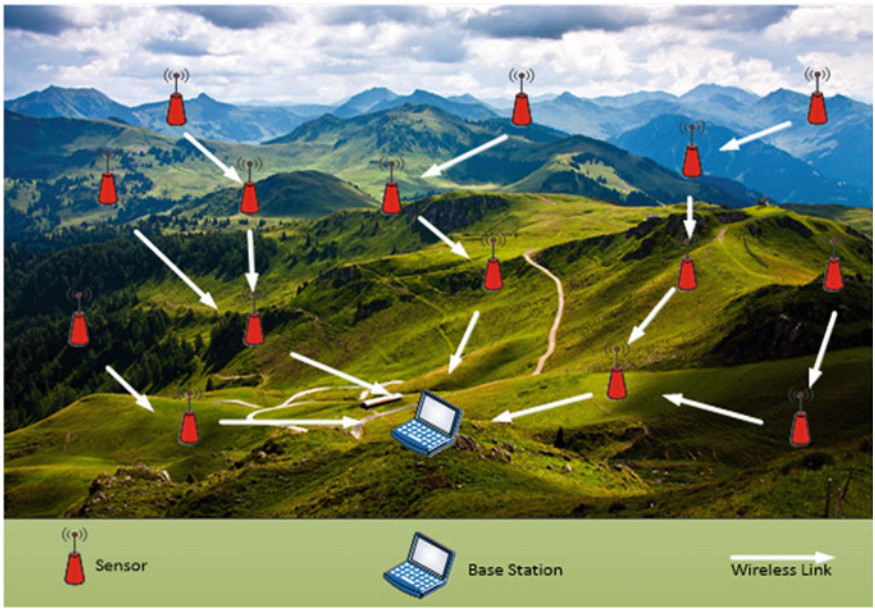


Fig. 1 An example of a wireless sensor network topology

replace nodes. In most of these applications, the failure of the BS is the most noteworthy since it would disrupt the network operation and often makes WSN useless. In addition, the failure of the BS can cause loss of important data that may not have been sent to the remote users. Hence, the unique role of the BS attracts the adversary's attention. An adversary can nullify the value of WSN by simply disrupting or physically damaging the BS. To do so, the adversary should first distinguish the BS among the other nodes in the network and determine its location. To successfully protect the BS from a nefarious adversary, we must keep its role, identity, and location anonymous. On the other hand, in some applications, the critical nodes are the sources rather than the destination of data. For example, an adversary opts to find the sensor that is reporting on the status of important assets. In these scenarios, protecting the asset's location from an external observer becomes critical.

As described earlier, data packets are forwarded from sensor nodes toward the BS through multi-hop routing in order to conserve energy. Sensor nodes may generate packets periodically or when they detect some event in their surroundings. In both operational models, the BS's location can be exposed by the high transmission rate in the BS's vicinity where the data routes merge (since the BS is a sink of all data traffic). On the other hand, the location of a data source cannot be exposed if all sensor nodes transmit packets periodically. Therefore, the problem of source location privacy exists only when sensor nodes generate packets to report the presence of an event and its activity. A passive adversary silently intercepts transmissions and applies traffic analysis techniques to infer properties about the structure of the network topology and the profile of the various nodes.

Deng et al. [4] have identified two classes of passive traffic analysis attacks that can be applied to WSNs. The first is the rate monitoring attack, where an attacker tracks the packet transmission rate. A passive eavesdropper can determine the location of the BS by identifying the nodes that transmit at high frequency. Similarly, the adversary can locate the source of a transmission due to the large volume of packets originating from a small area. The second and most sophisticated is the time correlation (or statistical correlation) attack, where an attacker determines the correlation in packets transmission time between neighboring nodes. The node that serves as a relay on a data route is expected to transmit a packet soon after receiving it. Thus, by accumulating observations, the attacker can correlate sender–receiver pairs and infer the routing paths that lead to the BS or locate the source of a transmission by back-tracing the hop-by-hop progression of packets generated at the source node due to a detected event. While packet encryption and anonymous routing are usually used to hide the identity of the BS and sensor nodes [5], the radio transmission of packets itself can be the base for traffic analysis by an adversary in order to expose the location privacy of WSN [6, 7, 8].

Unless traffic analysis countermeasures are being applied to WSN, an adversary can identify a target entity by analyzing the traffic patterns. To increase the location privacy of WSN, a number of anonymity boosting techniques have been proposed in the literature [9–30, 31–56]. The traffic analysis countermeasures usually aim to attract adversary's attention far from the BS/source location by altering the network's traffic pattern and consequently make the traffic analysis inclusive.

However, anonymity boosting solutions usually impose overhead on the individual nodes; therefore, protecting the WSN assets is subject to trade-off so that the network's performance and lifetime are not sacrificed for the sake of anonymity. In addition, contemporary attack models found in the literature do make assumptions that degrade the adversary's capabilities and cause the used metrics to misrepresent the level of anonymity. This chapter opts to cover all these issues.

The rest of this chapter is organized as follows. Section 2 defines anonymity in WSNs. Section 3 briefly analyzes existing source location privacy attack models and summarizes published source location privacy-preserving techniques. In Sect. 4, we describe the existing sink location anonymity attack models and categorize published traffic analysis countermeasures. In Sect. 5, we analyze the strengths and weaknesses of the existing sink location privacy attack models. In Sect. 6, we present EARS, a traffic analysis attack model that increases adversary's confidence in locating the BS while reducing the traffic analysis complexity. Finally, Sect. 7 concludes the chapter with a summary and a highlight of open research problems.

2 Anonymity Definition and Categorization

System's anonymity can be defined as the level of uncertainty an adversary has while seeking to identify a target entity within the system. Therefore, the anonymity of a system depends on the adversary's capabilities and the level of sophistication. An entity stays anonymous if it cannot be distinguished among others; thus, the basic prerequisite for providing anonymity to a particular entity is that there should be a set of entities $S_E(t)$ at a given time t with similar characteristics [13, 57]. From the adversary's perspective, the anonymity set $S_E(t)$ at given time t comprises all entities in the system that can be identified as a possible target. Therefore, at time t, the adversary can do no better than randomly selecting an entity $e \in S_E(t)$ as the target with probability $p_e = \frac{1}{|S_e(t)|}$. Over a period of time, the adversary gains further knowledge, e.g., by intercepting and analyzing transmitted packets, that helps in assigning a higher probability to a particular suspect e and hence increases confidence about e being the target.

In communication systems, the definition of anonymity refers to concealing the properties of the nodes that compose the network [13]. These properties can be the identity, location, or role of a node e within the network and thus lead to the following three types of anonymity:

1. *Identity Anonymity*: In a system that is composed from multiple entities, the need for identifying each entity by a unique name is necessary. For example, the Internet consists of hundreds of thousands of machines that are connected together, and each machine is identified by a static MAC address and a dynamic IP address. In many situations, it is necessary that an identity of a system component, e.g., a node in a WSN, must not be revealed to external entities and in many cases

even internal entities. Thus, achieving identity anonymity requires preventing an adversary from obtaining the addresses associated with particular system entities.

2. *Location Anonymity*: An adversary may be interested in knowing the location of a particular entity and not its identity. Location anonymity is defined as concealing an entity's position. For example, the adversary may know the IP address of a node without being aware of its location. On the other hand, an adversary may employ traffic analysis techniques and estimate a node's position without determining its identity. Depending on the application requirements, it may be necessary to provide location or identity anonymity or both [13].
3. *Role Anonymity*: Role anonymity refers to hiding the particular role activities of nodes within the network. For example, a particular node may be a gateway to a larger network, or a BS that provides protocol and time synchronization, resource assignments, and service to external users. The role of some nodes is critical for the network's operation which elevates the need of such type of anonymity.

To support anonymity in WSN, the system should ensure confidentiality for the exchanged packets between nodes and must prevent traffic analysis attack models from revealing the location of the target entity. Confidentiality of exchanged packets is in essence required for the content privacy threat that can be associated with the two types of anonymity, namely identity and role anonymity. The adversary attempts to decode the packet content of the intercepted network traffic in order to uncover the sensed data, and infer the identities and roles of nodes. This privacy threat can be countered by applying packet encryption and anonymous routing protocols such as [58–60]. On the other hand, location anonymity is associated with the contextual privacy threat, where the adversary eavesdrops on radio transmissions and applies traffic analysis techniques in order to infer valuable traffic information, uncover the routing topology, and localize the target entity [32]. In the rest of this chapter, location anonymity (privacy) is considered as the primary type of anonymity that is applied to WSNs.

As pointed out earlier, a passive adversary silently intercepts transmissions, seeking to locate the target node by applying a traffic analysis technique. In the literature, three types of passive adversaries could be identified. The first reflects a local (in situ) adversary who intercepts transmissions that occur in a small part of the network. The second type indicates an adversary that has partial view to the network traffic by deploying monitoring devices at different observation points. Finally, the third type is a global adversary who eavesdrops on communications in the entire network's operation area. An adversary that has local or partial view of the network could apply traffic analysis in the intercepted packets and gradually move to other parts of the network until finally locating the target entity. The global adversary applies traffic analysis using the collected traffic information throughout the entire network.

3 Source Location Privacy in WSNs

The threat of traffic analysis against source location privacy has gained increased attention from the research community in recent years [61]. In some applications, the location of the source that senses an event/subject is of utmost importance and must be hidden from the adversary. The adversary analyzes the intercepted transmissions aiming to locate the node that reports on an event.

3.1 Attack Models for Source Location Privacy

Published source location privacy techniques have focused on a "Panda-Hunter" scenario, where the adversary is termed as an animal poacher who seeks to determine the location of a mobile, tagged panda [31–56]. These techniques are employed under the presence of different privacy attack models which can be classified into three categories, namely global adversary, local adversary, and multi-local adversary [46].

Global adversary: The global adversary-based schemes [18, 34, 36, 44, 56] assume that the adversary can intercept every radio transmission occurring in the network. The adversary can eventually locate the source of an event notification by observing the originator of the traffic. To enhance source node's location privacy, some selected or all nodes should generate packets periodically, e.g., at fixed time intervals, in order to mimic the presence of multiple sources. If a selected node does not detect event's presence/activity at one time period, it generates a dummy packet, so that the adversary cannot know whether the packet is due to a real event or dummy data. It is obvious that in order to boost source location privacy against a global adversary model, a countermeasure should introduce significant overhead which degrades network performance and lifetime. Another reason for doubting the practicality of this attack model is that if the adversary has a global view to the network traffic, the source can be located without applying any traffic analysis technique [46].

Local adversary: This attack model assumes an in situ adversary with limited eavesdropping capabilities. As illustrated in Fig. 2, a local adversary is able to monitor only one region within the area of interest at a particular time. The adversary starts from the BS and tries to locate the origin of a transmission by back-tracing the hop-by-hop movement of the packets until reaching the source node. To elaborate, initially the adversary is positioned close to the BS and silently eavesdrops on communications. When the adversary intercepts a transmission made from node X_3, it moves to where X_3 is and waits. Similarly, the adversary moves closer to the source when node X_2 transmits a packet. Eventually, the adversary reaches the source node when node X_1 generates a new packet in order to report on the event. It has been shown in [32] that the wider the adversary's overhearing range is compared to the sensor node's transmission range, the higher the likelihood of locating the source becomes. Additionally, for events occurring in one location for some time period, the adversary

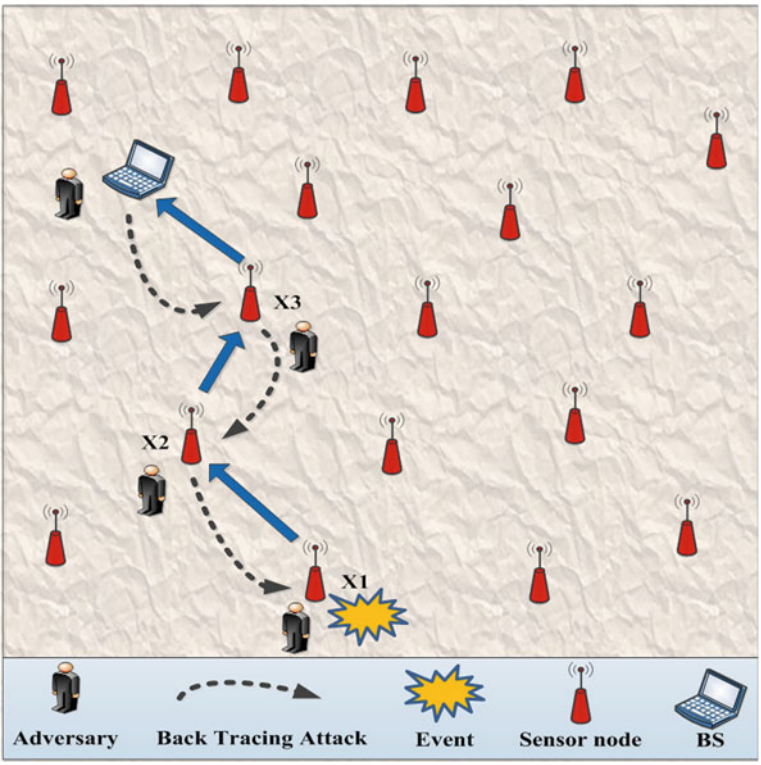


Fig. 2 Illustration of how a local adversary determines the data source

may capture a sufficient number of packets to locate the source area even if different routes are used to forward the data toward the BS.

Multi-Local adversary: In this attack model, the adversary is assumed to have a partial view to the network traffic [46] by distributing a group of monitoring devices at different observation points, as shown in Fig. 3. Traffic information, including the coordinates of the sender and the time of a packet transmission, is collected by the monitoring devices. The high generation rate of packets originating from small regions (source area) creates inconsistency in the network traffic pattern. Traffic flows are usually initiated from the sensor nodes within and around the source area due to event's presence/activities for some period of time. The adversary tries to identify the originator of the traffic by analyzing the collected data from the observation points through traffic analysis techniques such as nodes' transmission rate and packets' time correlation. Actually, the adversary uses traffic analysis back-tracing to move from the BS to the source. Basically, the continuous packet flow from the source area toward the BS causes some nodes on the routes to experience higher transmission rates than nodes far from the source. The adversary changes the observation points by moving the monitoring devices to more promising regions that can lead to the

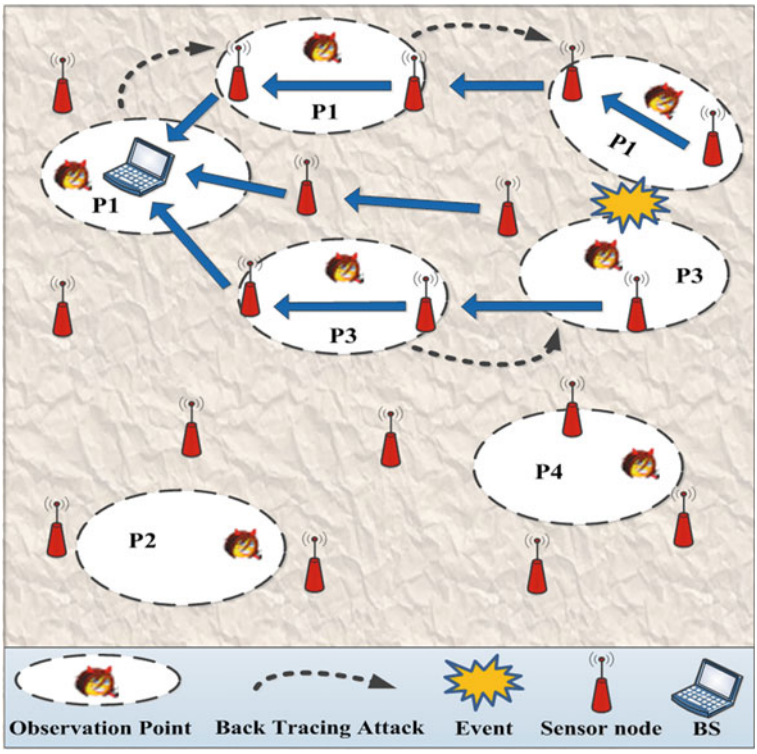


Fig. 3 A Multi-local adversary with four observation points

source area. The adversary suspects the presence of the source once a large drop in the packet sending rates is observed after passing the source's area. In other words, a possible source is identified by the adversary when the outgoing packet flows from a region are much more than the ingoing packet flows to that region.

3.2 Privacy-Preserving Measures for Source Location

Source location privacy measures focus on how information can be reliably and efficiently delivered to the sink while preventing the adversary from uncovering the location of the source. To enhance source location privacy, numerous anti-traffic analysis techniques have been proposed in the literature. Conti et al. [61] classify published techniques into eleven different categories, namely the random walk, geographic routing, delay, using dummy/fake data sources, cyclic entrapment, location anonymization, cross-layer routing, separate path routing, network coding, limiting the node detectability, and others. Overall, these techniques aim to protect the

source by employing various traffic pattern alteration approaches like dummy packets, dynamic route setup, packet flooding, mobile sinks, and data aggregation.

In the random walk-based methods proposed in [31, 33], generated packets follow a random path from the source to the sink. Therefore, it is difficult for an adversary to back-trace a packet since there are no established data routing paths. Other proposed solutions apply techniques that have similar effects, namely routing through randomly selected intermediate node/s [35, 40] and phantom routing [32]. In [38], the authors have studied the effect of intermediate nodes on anonymity and recommended selecting them closer to the sink area. This methodology is called the sink toroidal region (STaR). The separate path routing techniques described in [41] make sure that packets generated from the same source are routed to the sink through different nonintersecting paths. Meanwhile, the approach of [52] uses mobile sinks to collect the data in order to vary the routing paths based on the sink's new location. Spachos et al. [37] introduce opportunistic routing where the data paths are determined based on the proximity of relay nodes to the sink and on whether a node has recently served as a relay node. The objective is to vary the packet flow over time. The above techniques increase source location privacy by delaying the adversary progress toward the data source in hope of degrading adversary's confidence in whether the right strategy is being pursued. Figure 4 shows the effect of using various routing paths from the source to the destination. In this example, the adversary waits close to the BS until he captures the first packet that is routed to the BS through "Route 1". Then, the adversary moves where node "X1" is and waits for the next incoming packet in order to move closer to the source. Since different routes are used from the source to the BS, the adversary keeps waiting at node X1 until "Route 1" is used again. That way the adversary gets delayed and probably assumes that the intercepted transmission from node "X1" was a dummy packet.

Using fake data sources techniques introduces deceptive packets in order to alter the traffic pattern in the network [36, 42–47, 56]. The adversary is assumed to be incapable of differentiating the real from the fake packets. Chen and Lou [36] combine random walk, intermediate nodes, and deceptive packets. The presence of deceptive traffic complicates the local adversary's analysis since it is not possible to differentiate between real and fake packets. Therefore, the fake packets approach is probably the most effective in boosting the location privacy if applied in a sophisticated way. For that reason, in [44], two dummy traffic filtering techniques are presented. The first is the proxy-based filtering scheme (PFS) where proxy nodes are responsible to filter out dummy packets. Each proxy is connected to a set of nodes. The nodes send their packets (real or fake) to their associated proxy node. Proxy nodes filter out fake packets and queue the real packets. Finally, proxy nodes forward the queued real packets toward the sink or transmit a fake packet in case where the queue is empty. The second solution is the tree-based filtering scheme (TFS) where the proxies are organized into a tree hierarchy. In TFS, proxies are responsible not only for filtering out fake packets and forward real packets but also to aggregate real data. Additionally, proxies closer to the BS filter traffic from proxies that are farther. That way, TFS reduces the fake traffic and conserves energy. Finally, the optimal filtering scheme

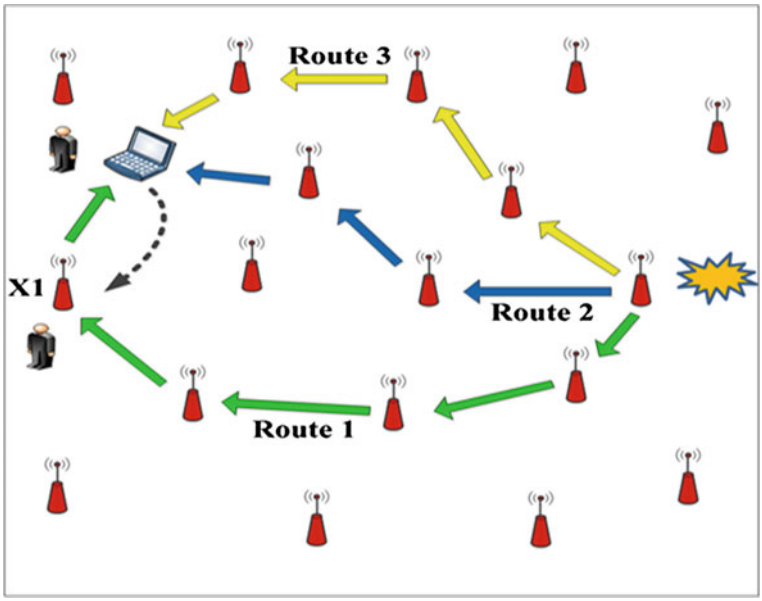


Fig. 4 Increasing source location privacy by using dynamic routing

(OFS) is presented in [47]. In OFS, every node can become a proxy to allow flexible routing and filtering so that the load is balanced and the WSN lifetime is extended.

Meanwhile, the cloud-based scheme [46] opts to blend real and fake packets between sources and intermediate nodes in order to create "clouds of traffic" that distract the adversary's attention while the intermediate node forwards the real packet to the BS. In the fitted probabilistic rate (FitProbRate) approach [56], nodes generate fake packets according to the statistical distribution of the real events. Real packets are embedded in the stream of fake packets, making it hard for the adversary to differentiate them. The idea of FitProbRate is extended in [45]. Essentially, when there is no real event detected in the monitoring area, nodes mimic the presence of a real event by initiating fake traffic. On the other hand, the cyclic entrapment techniques [42, 43] aim to disrupt the adversary's traffic analysis by creating cyclic routes of deceptive packets along the real path. An example is shown in Fig. 5; the red arrows represent the routes formed from fake packet transmissions, while the blue arrows represent the real path. That way, the local adversary gets trapped in the deceptive packets' routing circles without finding the real source. Mehta et al. [18] consider a global adversary and propose a solution called periodic collection (PeCo). In PeCo, every node has a decrementing transmission timer. When the timer expires, a node transmits the first packet in its queue. If there are no queued data packets at that time, the node simply transmits a fake packet. Fake packets are being dropped by the receiving nodes.

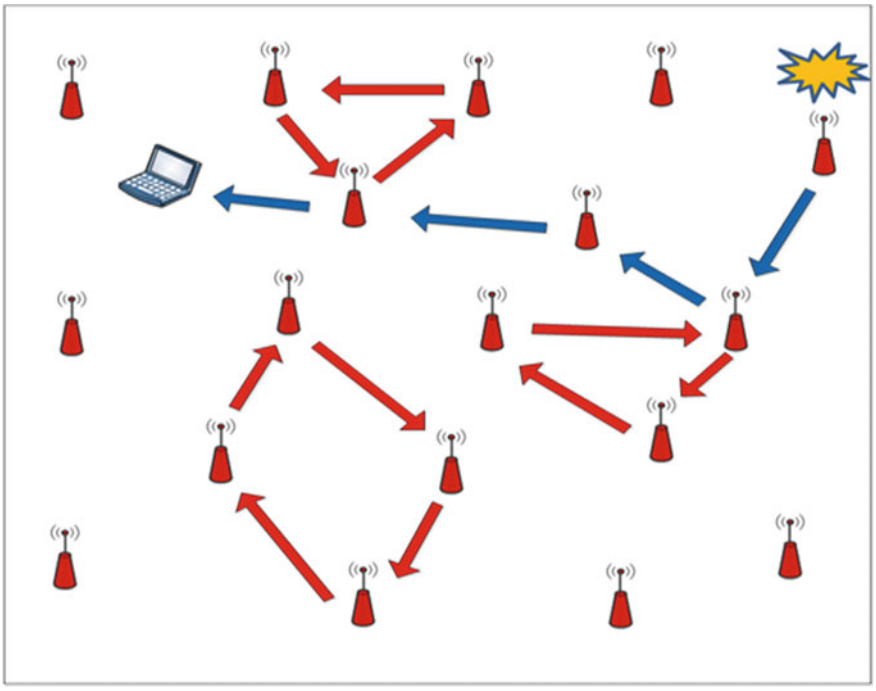


Fig. 5 Illustration of the cyclic entrapment technique for protecting the location privacy of data sources

Yang and Zhu [48] introduce an aggregation-based scheme that combines dummy packets and data aggregation to provide source location privacy. Other data aggregation approaches [39, 49–51] generally enable trade-off between location privacy and the increased overhead. On the other hand, flooding techniques have received a lot of attention in the literature despite their resource blindness. In [31], the authors describe the baseline flooding, where each node that receives a packet simply rebroadcasts it to neighbors. Flooding significantly increases the energy consumption across the network. Therefore, Kamat et al. [32] have proposed the flooding with forward probability technique where nodes randomly select some of their neighbors to multi-cast the packets to. Generally, flooding techniques do not significantly boost the source location privacy since the adversary can still back-trace a path to the source.

In the cross-layer approach and the double cross-layer approach [55], the nodes use the link-layer-based beacon frames, which are used for network maintenance, to share information on sensed events. Thus, a local adversary that eavesdrops on the packets (network layer) would miss part of the exchanged information and consequently does not find the real source. Oh and Gruteser [53] propose a multi-cooperator power control solution. Nodes collaboratively jam transmissions in order to complicate the adversarial transmission localization methods and thus make it harder for the adversary to find the actual source of the packet. Nonetheless, it is assumed that nodes

Table 1 Comparative summary of the source location privacy boosting techniques

Approach	Reference	Layer	Shortcomings
Dynamic routing	[31–33, 35–38, 40, 41, 46]	Network	Little effectiveness
Mobile sinks	[52]	Network	High frequency of topology changes
Dummy packets	[18, 42–48, 56]	Network	Increased overhead
Packet flooding	[31, 32]	Network	High overhead, little effectiveness
Data aggregation	[39, 44, 48–51]	Network	Little effectiveness
Cross-layer techniques	[55]	Cross	Little effectiveness
Transmission range reduction	[54]	Link	Little effectiveness
Multi-cooperator power control	[53]	Physical	High overhead and need for tight clock synchronization

will be able to detect the presence and localize the adversary. Finally, Tavli et al. [54] discuss how to increase source location privacy by lowering the radio transmission power. They point out that such a technique leads to increased forwarding packets and affects network's performance. Table 1 summarizes the source privacy-preserving techniques described above by comparing them based on the traffic pattern alteration approach and the network layer at which they are being applied.

4 Sink Location Privacy in WSN

Although some work, e.g., [36, 62], has stressed the importance of simultaneously providing both source and sink anonymity, source location privacy boosting techniques assume that the location of the BS (sink) is known to the adversary. From the adversary's perspective, it could be much harder or even impossible to locate the source if it does not locate the BS first. Precautionary measures can be employed in the design and the operation of the BS in order to avoid exposing it to the adversary. For example, the BS maintains a transmission power level equivalent to other nodes in WSN and limits its involvement in controlling traffic in order to be undistinguishable from other nodes in the network. Additionally, physical camouflage techniques may be applied to the BS such that an adversary cannot identify it visually, even while being present in the vicinity of the BS [18]. These techniques are very popular in military applications where equipments are blended with the environment or hidden to be out of sight. Therefore, trying to hide the location of the BS from an adversary that analyzes the traffic pattern of the network can be crucial even if the adversary

is seeking to locate the source of the traffic. For that reason, we focus in the rest of this chapter on the threat that an adversary can impose on the BS location privacy.

4.1 BS Location Privacy Attack Models

Three traffic analysis attack models are widely used in the literature, namely GSAT test, entropy, and evidence theory [13]. These attack models are in published work while developing countermeasures and validating their effectiveness in boosting the BS location privacy. In all the three attack models, the adversary eavesdrops on communications and applies techniques such as angle of arrival (AOA) and received signal strength (RSS), along with trilateration, to locate the source of the wireless transmission with considerable accuracy [63–68]. When a sensor node transmits a packet with a certain radio range, any of the nodes that lie within this range can be potential receivers of the packet. Due to insufficient localization accuracy, the adversary cannot accurately determine the location of a source and subsequently the potential receivers. To overcome that problem, the adversary maps the deployment area into a grid of equal-size square-shaped cells and determines the source cell of each transmission in the network. Thus, for each intercepted transmission, the adversary identifies the set of cells in which the potential receivers are located.

GSAT Test: The general satisfiability (GSAT) test is intended to measure the ability of a countermeasure to guard the network against an adversary that seeks to locate the BS. The GSAT test is motivated by the GSAT algorithm [69], which was proposed for solving NP-hard satisfiability problems. The GSAT test falls under the rate monitoring attack model and was first proposed by Deng et al. [4] as a tool of measuring the average number of steps an adversary makes until discovering the BS's cell. The GSAT test assumes a local eavesdropper rather than the more sophisticated global eavesdropper. An adversary that uses the GSAT algorithm performs greedy local search by identifying radio transmission hot spots and gradually moves to the area where the BS is located. As we mentioned earlier, nodes closer to the BS experience higher transmission ratio. Initially, the adversary starts from a random cell and monitors radio transmission activities within its vicinity, which includes its own cell and all the neighboring cells, for a certain period of time. Then the adversary moves to the neighboring cell with the highest transmission activities hoping to find the BS. If the adversary is already in the cell with the highest transmission activities among its neighbors and such a cell does not contain the BS, the adversary is said to be trapped in a local maxima. In such a case, the adversary randomly chooses a neighboring cell to move to. The greedy search terminates when the BS cell is reached. The number of moves that the adversary takes until reaching the BS represents the GSAT number. The more the network traffic is evenly distributed, the higher GSAT number the adversary gets.

Entropy: The entropy model uses Shannon's information theory [70] as a mathematical foundation. Diaz et al. [71] and Serjantov and Danezis [72] proposed entropy for gauging the anonymity of a system. Basically, sensor nodes are spread throughout

the deployment area of a WSN. A global adversary with passive presence divides the entire deployment area into a grid of N^2 square cells and tries to locate the BS's cell. The adversary assigns to each cell a probability that the BS is located within that cell. Let p_i be the probability at time t that a cell i contains the BS. Then the entropy of the system at time t is defined as:

$$entr_t = -\sum_{i=0}^{N^2-1} \left[p_i \times \log_2 p_i \right] \tag{4.1}$$

When the adversary starts eavesdropping on the network, each cell is assigned an equal probability of hosting the BS: i.e., the probability for each cell will be $\frac{1}{N^2}$. Hence, the initial entropy of the system will be $entr_0 = \log_2 N^2$ which reflects the maximum entropy. An adversary learns new information and adjusts the cells' probabilities by analyzing the traffic. The information that an adversary has learnt at time t can be expressed as $entr_0 - entr_t$. The ratio $d = \frac{entr_t}{entr_0}$ defines the system's anonymity at time t. When the adversary successfully identifies the BS, the probability of its cell becomes 1, the entropy becomes 0, and subsequently the anonymity will be zero at that time. Ideally the anonymity is 1 and stays high when the traffic is evenly distributed across the network. Thus, in the entropy attack model, an adversary tries to uncover the BS's location by identifying the cell where the largest traffic volume is occurring and concludes that the sink should be located within that cell [17, 22, 23]. The traffic volume (TV) of a cell i can be computed as follows:

$$TV(i) = num.of\ tranmissions \times transm.range \times packet\ size \tag{4.2}$$

Assuming that the transmission range is the same for all sensor nodes and the packet size is the same for any type of packet, then the TV of a cell can be reduced to the number of packet transmissions occurring within that cell. The probability p_i in Eq. (4.1) that an adversary assigns to each cell can be defined as the TV measured by the adversary for cell i divided by the total TV across all cells.

Figure 6 shows an example to illustrate this attack model. The green arrows represent the routing topology, red antennas denote the sensor nodes, and the blue computer represents the BS. For this example, each node generates 15 packets and is responsible for forwarding the generated and incoming packets to the next hop node in the routing path. The total number of transmissions each node makes (by accumulating the generated and forwarded packets) is shown next to the outgoing arrows from each node. The cell dimension is assumed to be equal to the nodes' transmission range, dividing the deployment area into a 3×3 grid. The "simplified" TV (assuming equal transmission range and packet size for each node) of each cell is shown in the yellow box of each cell. The BS's cell has the highest TV among the cells. In general, cells closer to the BS have higher TV and subsequently higher p_i. This also applies to the BS's cell itself if its size is large and has a considerable number of sensors.

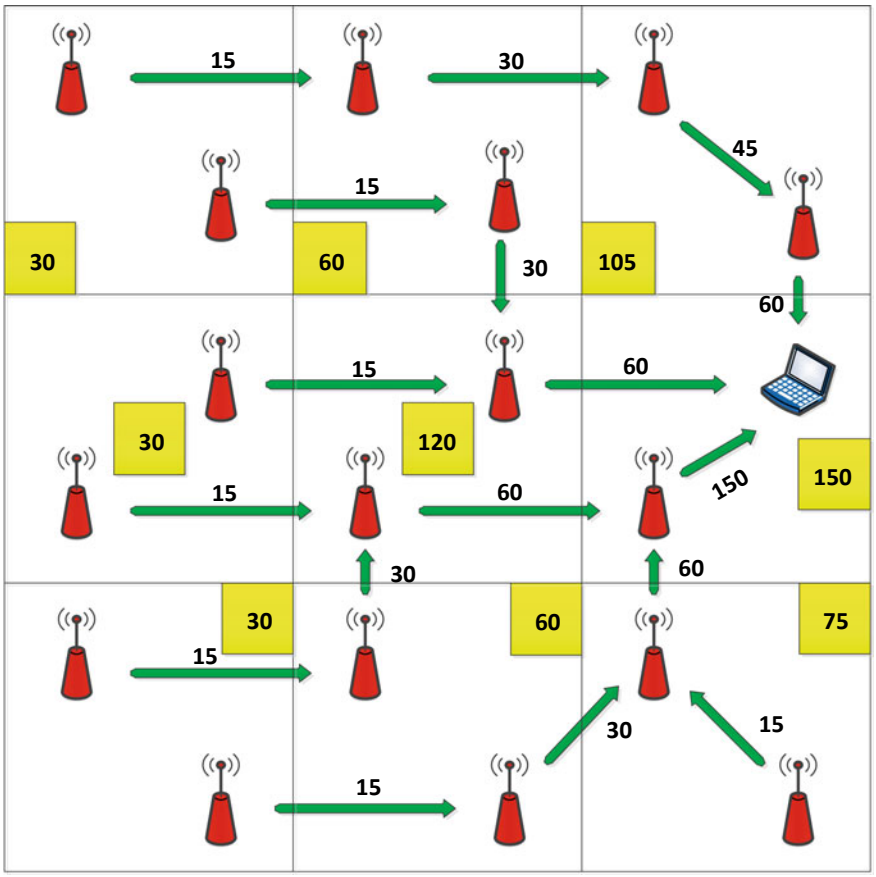


Fig. 6 Illustration of how TV is applied through an example scenario

Evidence Theory: Huang [73] first proposed evidence theory (ET) as a network traffic analysis model for measuring anonymity. ET falls under the statistical correlation attack model and has received attention from researchers because it provides a global adversary analysis framework that identifies a node or a cell with the highest likelihood of being or containing the BS, respectively. In ET, the adversary observes the network over a period of time, aiming to implicate a certain node as the BS. The attacker correlates the intercepted transmissions to compose all routing paths by considering all possible recipients of a transmission. Since multi-hop routing is used for disseminating data to the BS, the closer a node is to the BS the higher transmission rate it experiences. Therefore, most of the adversary's composed paths merge in the BS's vicinity, revealing the BS's location.

Specifically, the global adversary silently eavesdrops on the radio communications in the observed area. Each time a packet is intercepted during transmission from a source to a destination, it is considered an evidence of a communication link between

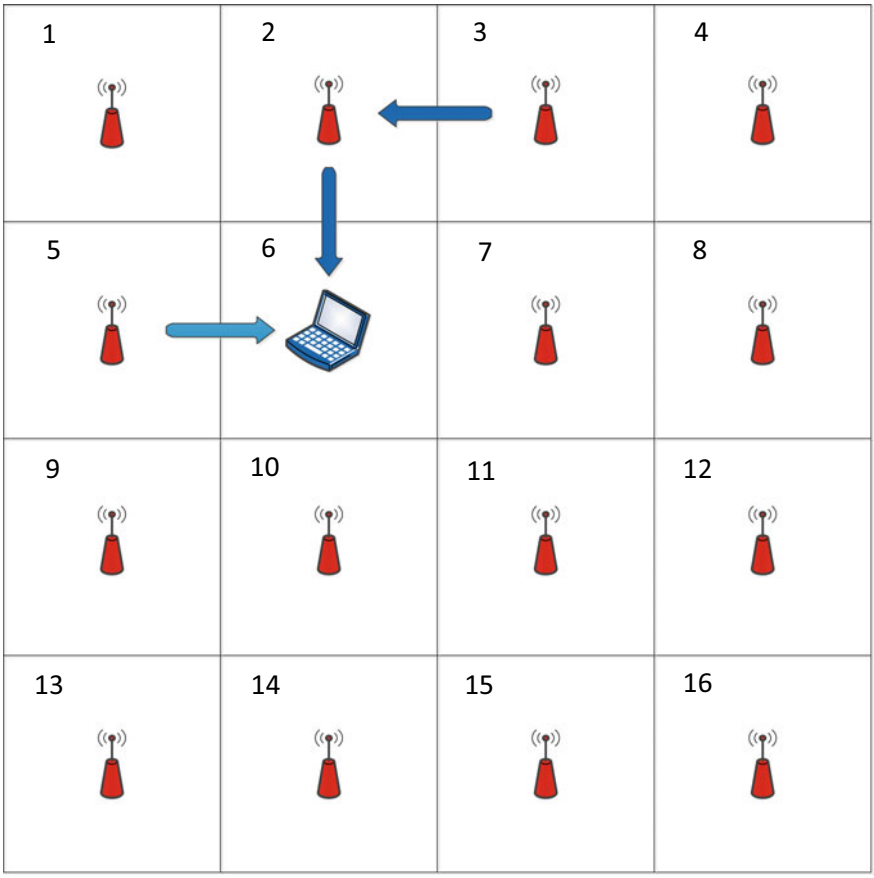


Fig. 7 Illustration of how ET is applied through an example scenario

source–destination pair. The approximate location of the source determines a set of probable receivers based on the estimated output power of the source radio. The adversary collects evidences and correlates them in order to deduce composite end-to-end communication paths. If S is the set of nodes in a network, the evidence E(R) for an end-to-end path R between two nodes x and $y \in$ S is defined as:

$$E(R) = min_{U \subseteq R}\{E(U)\}, \quad |R| \geq 2 \tag{4.3}$$

where $|R|$ represents the number of sensors that compose the inferred path. The normalized evidence of a path R is given by $m(R) = \frac{E(R)}{\sum E(V)}$, where V represents any possible composed path. Basically, $m(R)$ expresses the proportion to which all evidences collected from the adversary support the claim that a particular path R is part of the routing topology [73, 12, 13]. The weighted Belief, $Bel(y)$, denotes the

Table 2 Collected evidences and composed paths for the example scenario in Fig. 7

Trans. cell	Evidence	E(V)	M(V)
3	E(3, 2), E(3, 4), E(3, 6), E(3, 7), E(3, 8)	1	0.2
2	E(2, 1), E(2, 3), E(2, 5), E(2, 6), E(2, 7)	1	0.1
5	E(5, 1), E(5, 2), E(5, 6), E(5, 9), E(5, 10)	1	0.066
Derived	E(3, 2, 1), E(3, 2, 5), E(3, 2, 6), E(3, 2, 7), E(3, 2, 5, 1), E(3, 2, 5, 6), E(3, 2, 5, 9), E(3, 2, 5, 10), E(2, 5, 1), E(2, 5, 6), E(2, 5, 9), E(2, 5, 10)	1	0.037

adversary's confidence in the existence of paths Y that end at a specific node y and is defined as:

$$Bel(y) = \sum_{Y \subseteq V} n \times \sum_{U|U \subseteq Y} m(U) \tag{4.4}$$

where n denotes the length of the path Y. In few words, $Bel(y)$ refers to the sum of each Y's subpath's normalized evidences multiplied by the length of Y's path. A small weighted Belief value for the BS corresponds to decreased adversary confidence or higher BS anonymity, and vice versa.

As described earlier, the adversary experiences an error when localizing the source of a radio transmission and subsequently when determining the potential destinations. To overcome this problem, the adversary partitions the target WSN into a $N \times N$ grid. Consequently, y now refers to one of the N^2 cells within the adversary's grid and not to a particular node. Figure 7 shows an illustrative example of ET analysis applied to a 4×4 grid, where the cell side is equal to the transmission range of nodes. Two packets are generated from the nodes in cells 3 and 5, respectively. The BS is located in cell 6. The first packet is forwarded to the BS through the node in the second cell. If we assume that the time-based order of transmissions according to the cell number is 3, 2, and 5, then the adversary collects evidences and derives all possible paths in Table 2.

Most paths in Table 2 are ending at cell 6. Cell 6 has also the highest weighted Belief, that is: $(2 \times 0.037) + (2 \times 0.037) + (2 \times 0.037) + (3 \times (0.037 \times 3)) + (3 \times (0.037 \times 3)) + (4 \times (0.037 \times 6)) = 1.776$. As we can see in Table 2, the collected evidences derive composite paths based on the sequence of radio transmissions. The radio transmission order of the collected evidences enables the adversary to compose derived paths based on ordered pair-wise evidences. That way the derived paths are composed based on time-correlated transmissions (for this example, 3, 2, 5). If the adversary composes derived paths based on all possible permutations of the collected transmissions, then (i) the number of the derived paths increases significantly, in fact exponentially, and (ii) the weighted Belief value of each cell will not be valid.

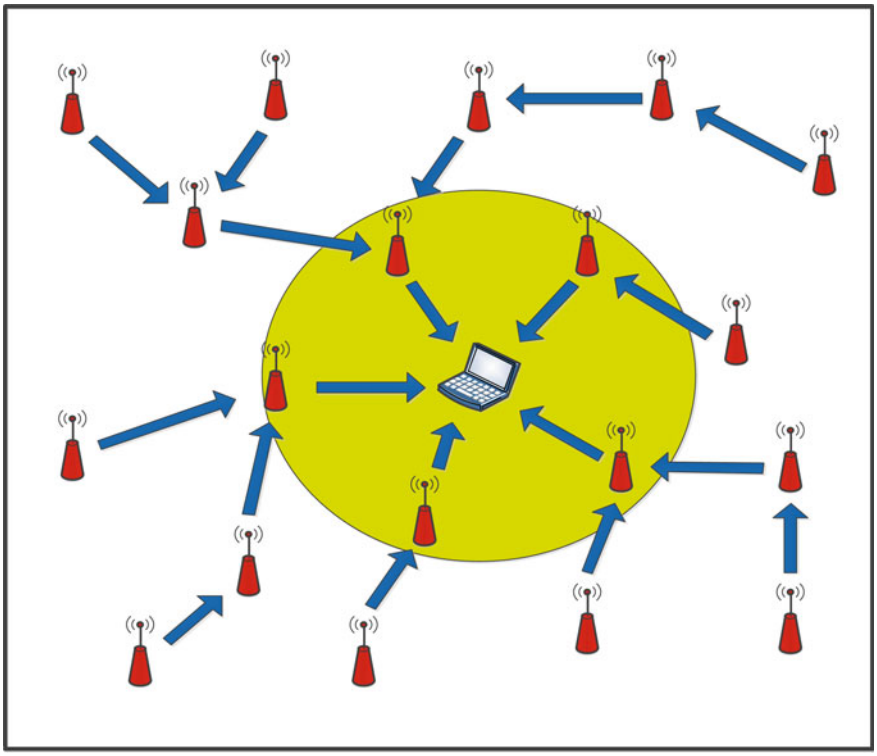


Fig. 8 A sample network where traffic flows (blue arrows) merge at BS vicinity (yellow circle)

4.2 Base Station Location Privacy-Preserving Measures

Traffic pattern alteration approaches, discussed earlier in Sect. 3, can successfully boost the source location privacy. While similar or same approaches can be applied in order to increase the BS location privacy, most of these approaches do not suffice for achieving that goal. There are two main reasons that make the transition of these approaches from source to sink location privacy insufficient. First, the BS location privacy boosting techniques should be able to defend against a global, more "sophisticated" adversary instead of a local or a partial one. Second and most importantly, when sensor nodes generate packets periodically, all the traffic flows merge in the BS vicinity, no matter the route a packet follows or if the BS is mobile or how many fake packets would be injected along each route, etc. Figure 8 shows how the traffic routes (blue arrows) merge into the BS vicinity (yellow circle). Even if the traffic routes vary over time while sensor nodes generate packets, the BS is exposed from the huge incoming traffic and cannot be hidden from an adversary without sacrificing network's resources and performance. Such trade-off between location privacy and network performance and lifetime should always be under consideration.

The crucial role of the BS has motivated the development of traffic analysis countermeasures in order to sustain the location privacy of the BS while considering the aforementioned trade-off. The objective of these countermeasures is to significantly alter the traffic pattern of the network in order to complicate the adversary's analysis, decrease its confident about locating the BS, and most importantly avert attention by exposing other network nodes/areas as the traffic's sink. To achieve that goal, traffic analysis countermeasures usually employ various approaches like dynamic routing, deceptive packets, data aggregation, packet flooding, mobile BS, and fake sinks. These approaches, as can be easily guessed, are imposing overhead. For that reason, cross-layer countermeasure techniques have been explored intending not only to increase the level of location privacy but also to conserve energy and maintain acceptable network performance.

Conner et al. [9] introduce a decoy BS that diverts the adversary's attention away from the real BS. All data packets are first forwarded to the dummy BS and then re-routed to the real BS. We illustrate that idea in Fig. 9, where all traffic is routed (blue arrows) to the decoy BS (blue sensor node) and then forwarded from the decoy sink to the BS (red arrows). Acharya and Younis [13] have proposed two techniques to boost the BS anonymity level. The first requires that the BS retransmits a subset of the packets it receives with different time-to-live in terms of the number of hops for which these packets are relayed. The second technique assumes a mobile BS that can relocate itself to a more secure location. In [30], the authors try to mimic the BS behavior by using aggregator nodes, delaying the real traffic, injecting deceptive packets, and varying the routing topology. Aggregator nodes (ANs) are responsible for collecting real packets and randomly forwarding them toward the BS via other AN. Real traffic is being delayed by increasing the time-to-live at each AN hop. ANs and regular sensor nodes introduce deceptive packets as well. Finally, the BS has the capability to update the routing topology at any time.

In ATA [20], each node generates the same TV with the sink's neighbors by transmitting deceptive packets so that the traffic density is evenly distributed across the network. In [14], data generated by each node is destined not only to the sink but also to every other node in the network (flooding) such that all nodes including the BS have equal share of total incoming and outgoing flows. Meanwhile, the technique proposed in [10, 11, 16] increases the BS location privacy by strategically injecting deceptive packets in areas with low traffic density in order to manipulate the traffic pattern that an eavesdropping adversary observes. By targeting only low-traffic areas, it is possible to boost the location privacy while conserving energy. The differential enforced fractal propagation (DEFP) technique proposed in [4] is a combination of multiple anti-traffic analysis approaches, namely multi-parent routing, random walk, and fractal propagation. Deceptive packets are propagated in the network. A node uses a lottery scheduling algorithm to choose one of its neighbors to forward the deceptive packet to. That way, fake packets form traffic patterns that turn specific areas into hot spots.

In [29], two different types of nodes are used in order to hide the BS. Nodes located near the BS are called BLAST nodes, while other nodes are just regular nodes. BLAST nodes have different communication ranges and forward the received

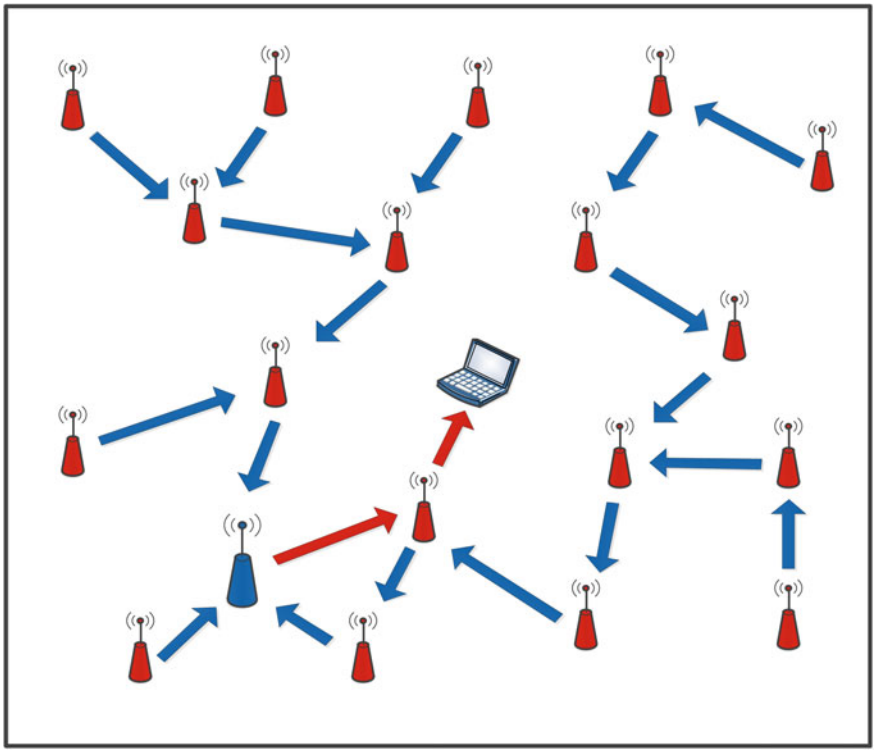


Fig. 9 Increase in BS location privacy by using a decoy sink (blue sensor node)

packets to the BS through shortest paths. Deceptive packets are used in order to distribute the traffic across the network. The approach of [15] opts to grow the computational complexity of an adversary's ET-based analysis by increasing the transmission power used by all network nodes. To provide an additional element of confusion, distributed beamforming is exploited as a cross-layer technique in [20, 21]. Basically, multiple nodes cooperatively transmit a packet at short range such that the individual transmissions sum up at a distant receiver. Thus, the transmissions would appear to the adversary as unrelated and would not implicate a sender–receiver relationship.

Data aggregation is exploited as a means for boosting the BS location anonymity in [26–28]. Selected cluster head (CH) nodes are responsible for collecting and aggregating real packets. Aggregated data are forwarded from CH to CH toward the BS. In [26], a Hamiltonian cycle is used as the routing topology among the CH and the BS so that the BS appears as a CH to an external observer. In [27], only one CH node is able to forward the aggregated data to the BS. To achieve that, CH nodes form a hierarchical routing topology in order to propagate the data to the sink. Furthermore, the BS selectively transmits the packets to one of the CHs so that it

appears as one of the regular network nodes rather than the sink of all traffic. Finally, CoDa [28] combines the data payload of multiple packets into a single packet in order to reduce the pair-wise evidences and cut down traffic flows that are headed toward the BS. An additional challenge to the adversary's traffic analysis is introduced by the forwarding delay that is accumulated since incoming packets should be queued in order to be combined.

Fake sinks and deceptive relay nodes have been pursued for boosting the location privacy of the BS [18, 22–24]. In [18], simulation is used to explore the effect of fake sinks; yet no algorithm was proposed to determine how many, where they have to be, and how to route traffic to them. Di Ying et al. [22] have proposed an algorithm to determine the location of fake sinks aiming to minimize the total traffic load in the network. However, the fact that the routing topology and the first fake sink are picked randomly cannot guarantee that the selected set of fake sinks achieve the desired goal. Additionally, nodes are assumed to be deployed on a grid topology, with the distance between two neighboring nodes on the grid equal to the communication range of nodes. Obviously such an assumption limits the applicability of the algorithm.

Both IATA [22] and MoRF [23] address the identification of the fake sinks; yet only MoRF factors in the traffic load by probabilistically generating deceptive packets toward the fake sinks such that the traffic is well distributed across the network. Figure 10 illustrates how MoRF alters the traffic pattern by routing deceptive packets to the fake sinks (blue sensor nodes). For each real packet that is generated and is being forwarded toward the BS (blue arrows), fake packets are probabilistically generated from nodes in the real path and forwarded (red arrows) toward each fake sink. While MSI [24] applies the same heuristic of MoRF to determine the fake sinks, unlike MoRF, MSI deterministically identifies deceptive relay nodes and defines their deceptive packet generation rate in order to turn the vicinity of the fake sinks into hot spots without a significant impact on the network's performance.

Table 3 summarizes the BS location privacy countermeasures described above and compares them based on the traffic pattern alteration approach and the network layer at which the countermeasure is being applied.

All countermeasures, described above, seek to enhance the BS location privacy by altering the traffic pattern of the network. As described earlier, the alteration of the network's traffic pattern decreases adversary's confidence in locating the BS and makes its analysis inclusive, but may not guarantee that the adversary would fail to discover the BS. Generally, we categorize the performance of a countermeasure based on the following criteria: (i) whether the countermeasure can resist traffic analysis attacks launched by a global attacker, or only a local one, (ii) whether the countermeasure is capable of defending the BS against both types of global adversary models: rate monitoring and statistical correlation attacks, (iii) whether the countermeasure allows balancing the interest in energy consumption and location privacy, and (iv) whether the countermeasure can be applied to any WSN topology.

Published countermeasures are usually evaluated while considering an adversary with specific capabilities. The fact that a countermeasure is evaluated against a local or global adversary that uses rate monitoring or statistical correlation attack model does not mean that this countermeasure cannot successfully protect the BS under

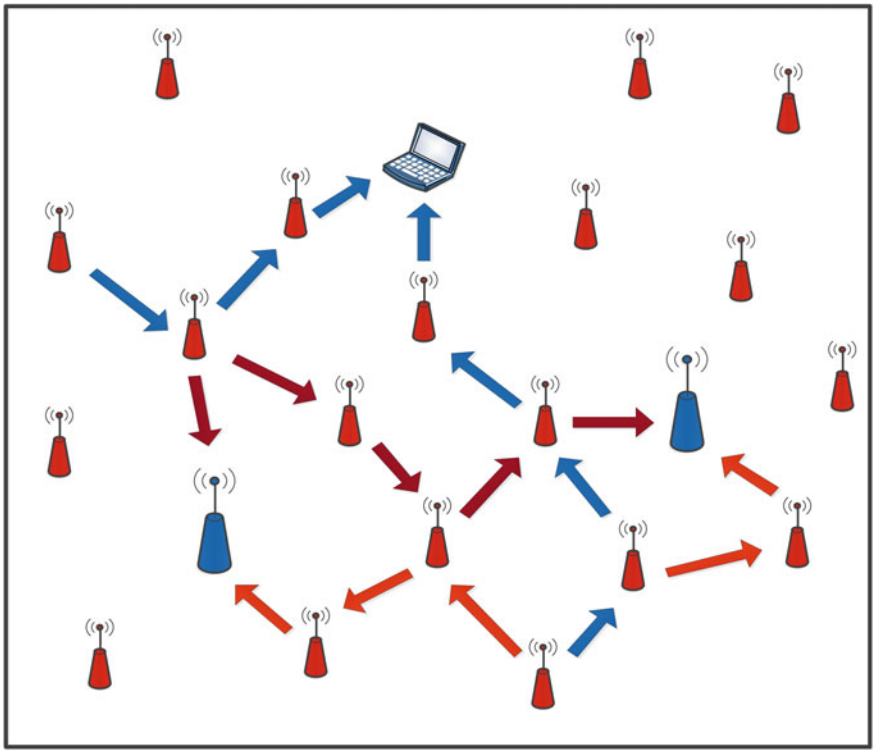


Fig. 10 Boosting the BS location privacy by routing deceptive packets (red arrows) toward each fake sink (blue sensor nodes)

different adversary models. Therefore, in order to assess whether a given countermeasure provides high location privacy level while satisfying (i) and (ii), the countermeasure should explore and efficiently address the following critical issues: (1) in which area/s of the network the traffic pattern should be altered, (2) how much the traffic pattern should be altered in these area/s, and (3) which nodes are responsible to accomplish the traffic pattern alteration. Tackling these issues should strike a balance between energy consumption and location privacy in order to satisfy (iii). Additionally, it would be beneficial if the mechanisms for addressing these issues vary over the operation time of WSN. Table 4 summarizes the capabilities of the discussed countermeasures to manage these conceptions.

Table 3 Comparative summary of the sink location privacy boosting techniques

Approach	Reference	Layer	Shortcomings
Dynamic routing	[4, 9, 30]	Network	Little effectiveness
Mobile sink/s	[13]	Network	High frequency of topology changes
Fake sinks	[9, 18, 22–24]	Network	Increased overhead
Deceptive packets	[4, 10, 11, 16–18, 22–24, 29, 30]	Network	Increased overhead
Packet flooding	[14]	Network	Excessive overhead
Data aggregation	[26, 28]	Network	Increased data delivery latency
Delay	[13, 30]	Network	Little effectiveness
Varying/growing transmission range	[15, 29]	Link	Increased medium access collision and signal interference
Distributed beamforming	[20, 21]	Physical	Need for tight clock synchronization

Table 4 Classifying the capabilities of published countermeasures for enhancing the BS location privacy

Criteria	Counter both types of global adversary models			Balancing privacy and network performance	Applied to any network topology
	Where?	How much?	Which nodes?		
[4]	✓	✓	✓	✓	✓
[9]			✓		✓
[11]			✓		✓
[13]	✓		✓		✓
[14]		✓	✓		✓
[15]					✓
[16]	✓	✓	✓		✓
[17]	✓	✓	✓		✓
[18]		✓	✓		✓
[20]		✓	✓	✓	✓
[21]		✓	✓	✓	✓
[22]	✓	✓	✓		
[23]	✓	✓	✓	✓	✓
[24]	✓	✓	✓	✓	✓
[26]			✓		✓
[27]			✓		✓
[28]			✓		✓
[29]			✓	✓	✓
[30]		✓		✓	✓

5 Sink Location Privacy Attack Models: Strengths and Weaknesses

As discussed in Sect. 4, three traffic analysis attack models, namely entropy, GSAT test, and evidence theory (ET), are widely used by published techniques to define countermeasures and to validate their effectiveness in boosting the BS anonymity. In Sect. 4.1, we used the traffic volume (TV) attack model to assess entropy, while detailed examples of how a global adversary applies ET or TV to discover the BS were also given. In this section, we analyze the strengths and weaknesses of these models from an adversary's point of view in order to assess whether existing countermeasures are as effective as they are thought of to be. Our analysis highlights some fundamental shortcomings in these models, and consequently, the effectiveness of the related countermeasures is degraded.

5.1 Traffic Volume—Discussion and Critique

In the traffic volume attack model, as was discussed in Sect. 4.1, the adversary is a global passive observer that eavesdrops on the entire deployment area and intercepts transmissions. The adversary maps the deployment area into a $N \times N$ grid of equal-size square-shaped cells and determines the source cell of each transmission in the network. To locate the BS, the adversary tries to identify the cell where the largest traffic volume is occurring. The traffic volume (TV) of each cell can be computed by using Eq. (4.2). Then the adversary assigns to each cell i a probability p_i for hosting the BS, which can be defined as the traffic volume measured by the adversary for cell i divided by the total traffic volume across all cells. The BS's cell and its neighboring cells are expected to have higher traffic volume and subsequently higher p_i.

Figures 11, 12, and 13 show an example of this attack model applied to different grid sizes. For simplicity, we assume that the transmission range is the same for all sensor nodes and the packet size is the same for all packets. Thus, the traffic volume of a cell in Eq. (4.2) can be reduced to the number of packet transmissions made within that cell. The green arrows reflect the routing topology, the red antennas represent the sensor nodes, and the blue computer denotes the BS. For this example, each node generates 15 packets and is responsible for forwarding the generated and incoming packets to the next hop node in the routing path. The number of transmissions each node makes by accumulating the generated and forwarded packets is shown next to the outgoing arrows from each node. In Fig. 11, the cell dimension is equal to the node's transmission range, dividing the deployment area into a 3×3 grid. The traffic volume of each cell is shown in the yellow box of each cell. The BS's cell has the highest traffic volume.

Figure 12 shows the effects of decreasing the cell size. Each cell contains at most one node, and the traffic volume of each cell is the number of packets that its nodes transmit. The BS's cell does not experience any transmission, and hence, its p_i will

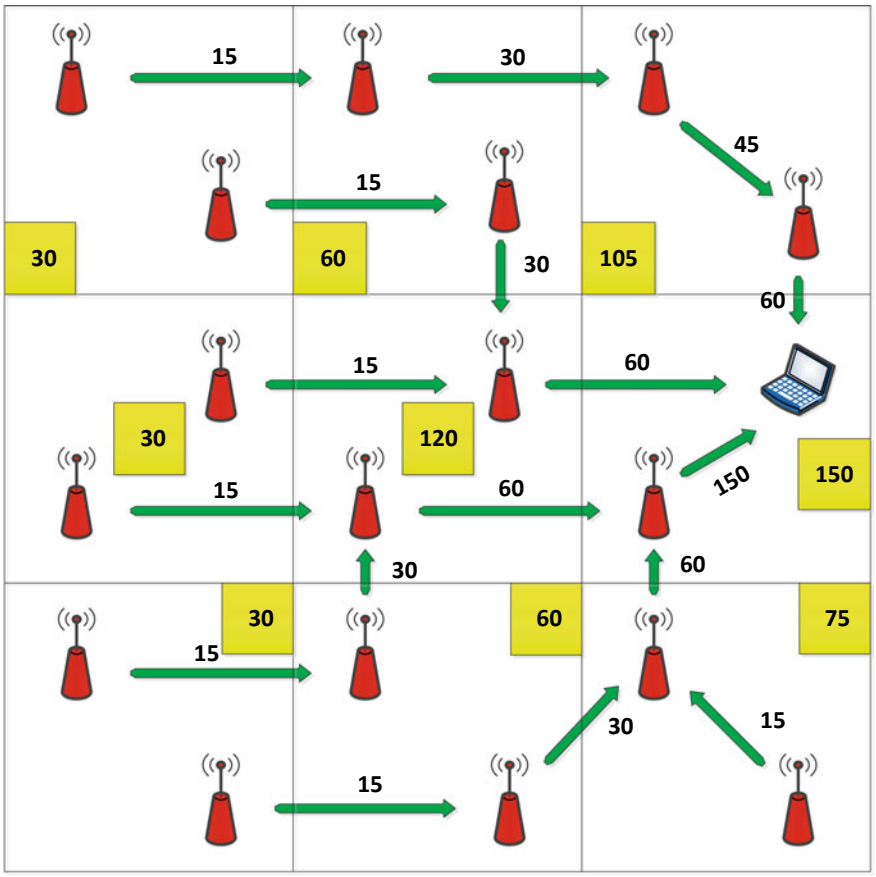


Fig. 11 Traffic volume distribution for the 3 × 3 grid

be zero. A close cell to the BS has the highest traffic volume, and subsequently, the adversary can conclude that the BS is located within that cell or in a neighboring cell otherwise. It is important to note that the communication range of nodes is still same.

Finally, in Fig. 13, we increase the cell dimensions, dividing the deployment area into a 2 × 2 grid. The cell in the bottom right corner experiences the highest traffic volume while the BS is located in another cell. Hence, increasing the cell size does not guarantee the highest p_i for the BS's cell. Basically, the routing topology and the distribution of the nodes in the deployment area significantly affect the traffic density.

A fundamental advantage of the TV analysis is its simplicity as its complexity is linear in the number of cells. Nonetheless, it suffers quite a few shortcomings. First, the probability p_i of a cell varies over time and depends on the traffic volume of that cell. If the transmission range is not the same for all nodes and/or the packet sizes

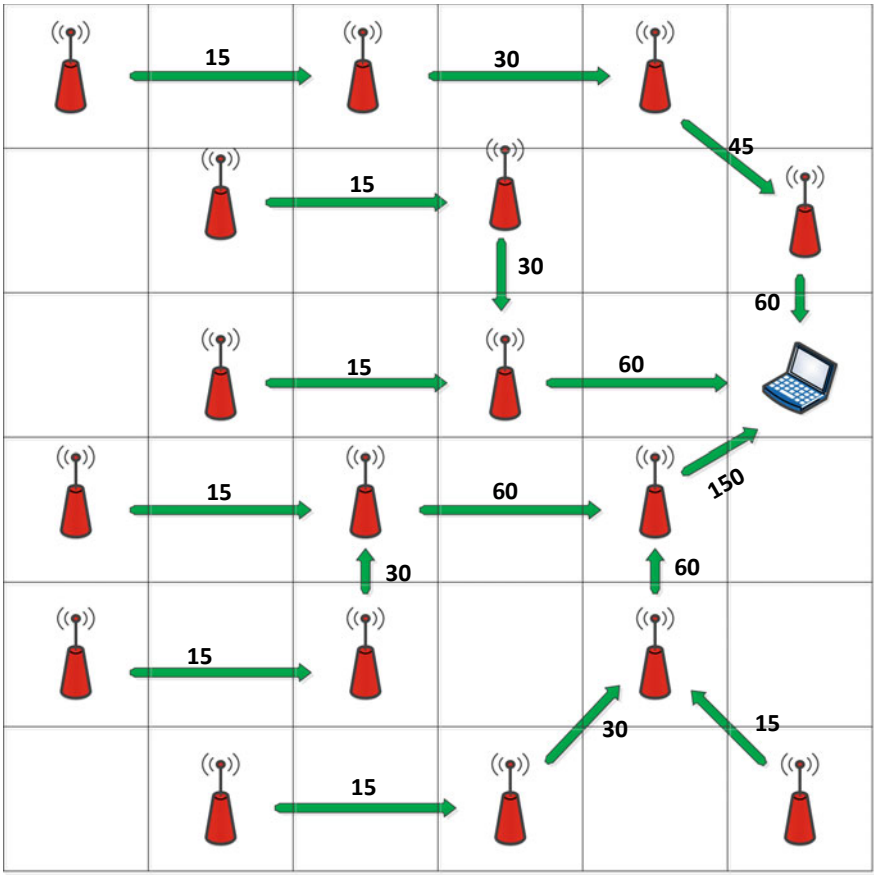


Fig. 12 Traffic volume distribution for the 6 × 6 grid

vary, then such an attack model could be easily deceived. Additionally, the cell size that the adversary considers can be too small such that most of the BS's neighboring nodes will belong to different cells. That results in assigning higher p_i to cells close to the BS than the BS's cell itself. Generally, the larger the cell size is, the higher the probability for identifying the BS cell by using this attack model becomes (not always though as we have shown above).

As a countermeasure, nodes located in cells that are far from the BS can insert redundant traffic and counter this attack model without significant effect on the network lifetime and packet delivery ratio. Hence, the traffic volume analysis could depend on the node distribution across the deployment area, cell size, transmission range of nodes, packet size, and redundant traffic. From the adversary's point of view, performing such an attack will require low computational power and will assign high probability to BS's cell if the cell size is relatively big. While TV is easy to be applied,

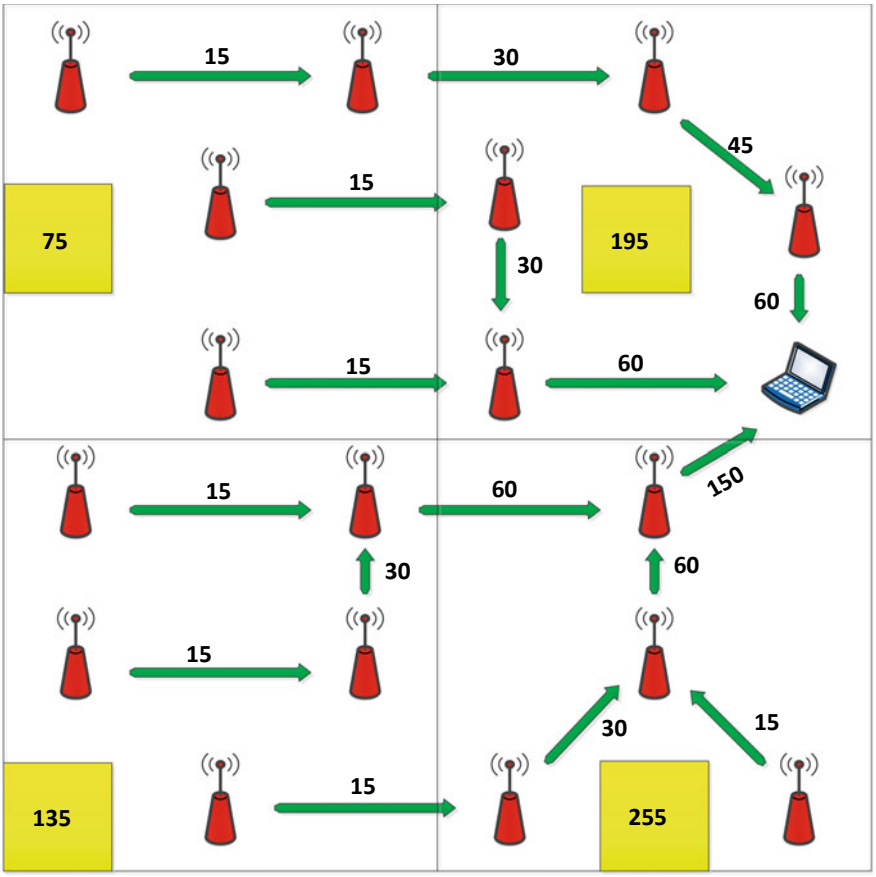


Fig. 13 Traffic volume distribution for the 2 × 2 grid

a sophisticated adversary knows that even a "simple" countermeasure technique that introduces redundant traffic far from the BS will defeat such an attack.

For that reason, the adversary should consider a cell that has higher p_i than a specific threshold as candidate to host the BS. Hence, the goal of the adversary is to identify a set K of cells that represent the set of possible BS locations. Thus, the probability of each candidate cell to host the BS is $1/|K|$, which reflects the level of uncertainty the attacker has in identifying the BS cell. Therefore, the uncertainty of the adversary depends on the threshold for p_i. If the threshold increases, the size of the candidate set decreases, but the adversary is not confident that the BS cell belongs to the candidate set. On the other hand, lowering the threshold boosts the adversary's confidence that the BS's cell belongs to the candidate set, yet the size of the candidate set is now larger.

5.2 GSAT Test—Discussion and Critique

The GSAT test assumes a local adversary rather than the more sophisticated global adversary. The adversary is assumed to divide the deployment area into a grid of equally sized square cells as before. The local adversary starts from a random cell i and eavesdrops on communications within its vicinity for a certain period of time. Then the adversary moves to the i's neighboring cell that experiences the highest transmission activities, hoping to find the BS. If no neighboring cell has more transmission count (local maxima), the adversary randomly chooses a neighboring cell at random and goes to. The GSAT number denotes the average number of moves before the adversary finds the BS.

Assuming that contemporary camouflaging techniques are employed to make it hard for the BS to be visually identified by the adversary, even if it is in the same cell, the GSAT test will never terminate. For this reason in [13], the adversary model is extended to track the total number of moves (cell transitions) and the number of times an individual cell is visited. At any time t, the GSAT score of a cell i is defined as:

$$G(i,\ t) = \frac{\#\ of\ cell\ visits}{\#\ of\ total\ moves}, \tag{5.1}$$

If an adversary has moved to one particular cell many times due to its high transmissions activities, it can be concluded with high confidence that the BS resides in that cell. Considering the previous example, we can see the effect of the cell size (Figs. 11, 12, 13) in this type of attack model. If the cell size is too small (Fig. 12), the GSAT score for the BS's cell will be too low and the number of moves taken by the adversary will not be less than a random search. The local view of the adversary, on the other hand, cannot assume big cell size. As it can be easily guessed, it is not difficult to counter the GSAT attack model; basically, nodes that are away from the BS can inject redundant traffic and boost the anonymity of the BS.

As discussed above, this attack model assumes a local adversary. An adversary with a global presence is more powerful. The GSAT score does not exclusively depend on the traffic distribution, but it also depends on the cell size (and subsequently the number of cells), the starting cell, the number of local maxima, and most importantly the randomness of the move the adversary takes to advance past a local maxima. Obviously, a traffic distribution with many local maxima can significantly delay the convergence of GSAT-based attacks.

5.3 Evidence Theory—Discussion and Critique

In Sect. 4.1, we have described the ET-based attack model. The ET attack model considers a more "sophisticated" global adversary, who passively eavesdrops on the radio communications in the observed area for a certain period of time. The

global adversary intercepts transmissions and correlates them in order to compose end-to-end communication paths. As described earlier, the adversary partitions the target WSN into a $N \times N$ grid, localizes the source of a transmitted packet, and determines the potential receivers. Then, the adversary computes the evidence $E(R)$, of each path R, by using Eq. (4.3). The normalized evidence of a path R is given by $m(R) = \frac{E(R)}{\sum E(V)}$, where V represents any possible composed path. Finally, the adversary, by using Eq. (4.4), computes the weighted Belief $Bel(y)$ for each cell y.

As discussed in [15], the complexity of discovering all possible paths using cell-based analysis is derived based on a breadth-search tree as $O(N^2 b^d)$, where b is the branching factor, d is the depth to search, and N^2 is the number of cells in a $N \times N$ grid. The branching factor b of a cell depends on the transmission range of the nodes within that cell. Assuming that all nodes have the same transmission range, the branching factor b will be the same for all cells and is defined by:

$$b = O\left(\sum_{i=1}^{q} 8 \times i\right), \tag{5.2}$$

where $q = \lceil Transmission\ Range / CellSide \rceil$. In essence, the parameter q determines the number of all possible recipient cells for a transmission made by a node given the radio range and cell dimensions. The depth to search d is $O(longest\ path)$. Since the longest path in a $N \times N$ grid could consist of N^2 cells, the complexity of the traffic analysis is represented as:

$$Complexity = N^2 \times O\left(\left(\sum_{i=1}^{q} 8 \times i\right)^{N^2}\right), \tag{5.3}$$

To reduce the complexity of the analysis, the adversary should reduce the branching factor. The minimum value for b is achieved when the cell side is equal or greater than the transmission range of nodes. Hence, decreasing the cell size can increase the complexity of ET traffic analysis. Additionally, the adversary can reduce the complexity by correlating evidences for short time windows, i.e., by considering the order of transmissions within a short duration so that a subset of the possible paths is excluded. The adversary computes the weighted Belief of each cell per time window based on the derived paths. All derived paths are deleted before the new time window begins. Finally, each cell is assigned the average of the cell's Belief values. In other words, the adversary reduces the depth to search d by deriving fewer composed paths with shorter length at the cost of lower confidence. As the time window expands, the analysis grows in complexity and the confidence of the adversary increases.

In Fig. 14, we use the example of Sect. 4.2 (Fig. 7), but now let us assume that the transmissions from nodes in cells 2 and 3 are captured within the same time window, while the transmission from cell 5 is captured during another time window. Table 5 shows the derived paths based on such an assumption. The number of paths is smaller and their length is shorter compared to the composed paths of Table 2 (Sect. 4.2).

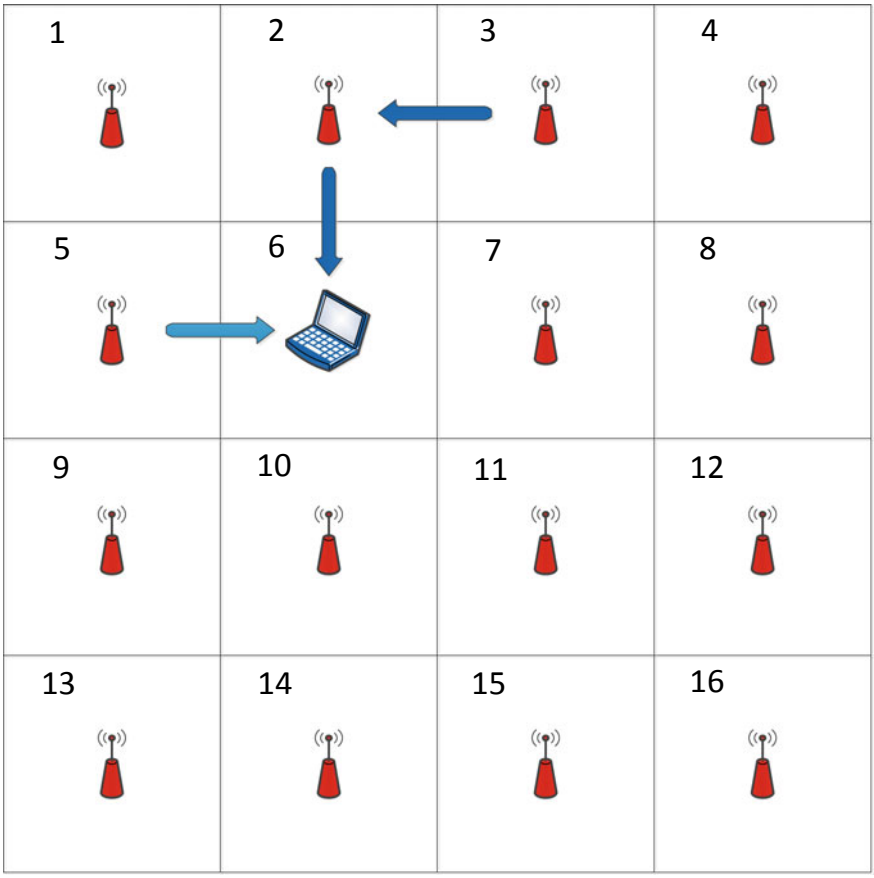


Fig. 14 Illustration of how ET is applied with time windows

The weighted Belief of cell 6 is $(0.07\times 2)+(0.07\times 2)+(3\times(0.07\times 3))=0.91$ for the first time window and 0.2 for the second time window. Cell 6 has an average weighted Belief equal to 0.555 that is still the maximum compared to the Belief of other cells, yet it is less than the Belief value that cell 6 had (1.776) before we apply a shorter time window (Sect. 4.1).

Consider now the example of Fig. 15, where the cell side equals the node's transmission range. Assume that each of the sensor nodes within cell 5 generates a packet. The arrows represent the routing paths. There are 3 transmitted packets that the adversary captures during the same time window. Assume that the dark blue path finishes before the light blue path starts. Table 6 shows the derived paths during this time window. Only two paths end at the BS's cell, and consequently, the BS's cell has lower Belief value than most of the cells. The reason is that the BS's cell does not behave as a sink anymore. The BS's cell experiences transmissions because of the sensor node within it. This attack model points out many reverse paths, i.e., going away

Table 5 Collected evidences and composed paths for the example in Fig. 14 while assuming a small time window

Time window	Trans. cell	Evidence	E(V)	M(V)
1	3	E(3, 2), E(3, 4), E(3, 6), E(3, 7), E(3, 8)	1	0.2
	2	E(2, 1), E(2, 3), E(2, 5), E(2, 6), E(2, 7)	1	0.1
	Derived	E(3, 2, 1), E(3, 2, 5), E(3, 2, 6), E(3, 2, 7)	1	0.07
2	5	E(5, 1), E(5, 2), E(5, 6), E(5, 9), E(5, 10)	1	0.2
	Derived	–	1	0.2

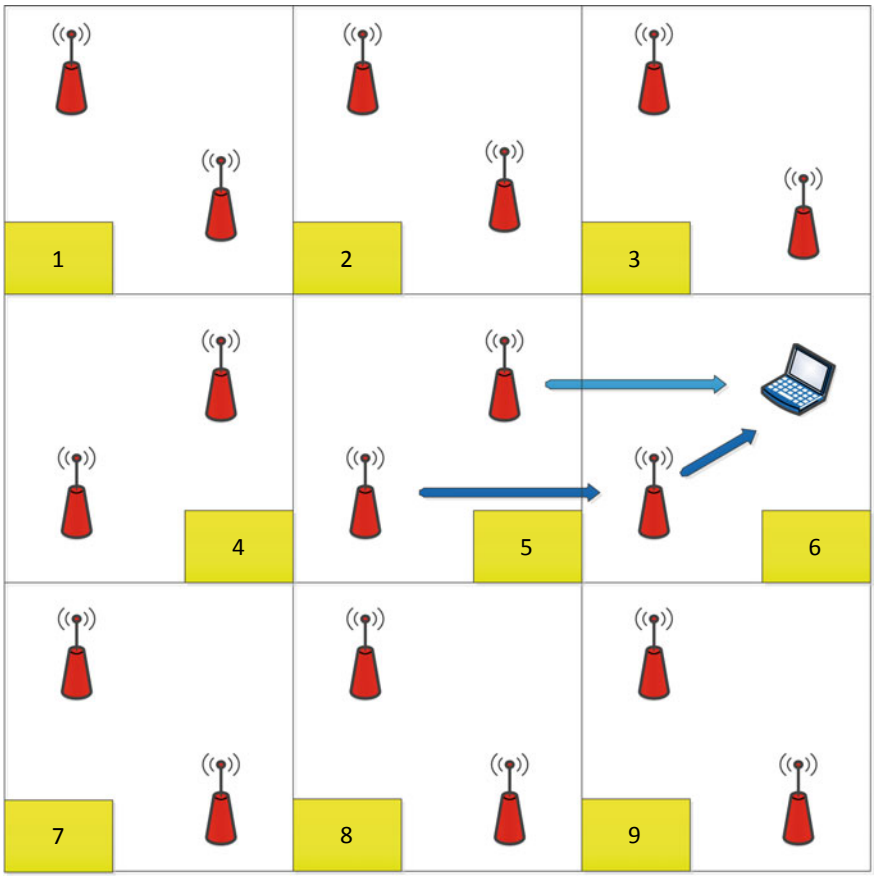


Fig. 15 An example scenario that illustrates how ET fails when BS's cell experiences transmissions

from the BS, if sensor nodes belong in the same cell with the BS. The more sensor nodes belong to the BS's cell, the more BS's cell weighted Belief value diminishes.

Table 6 Derived paths for the collected evidences for the example in Fig. 15

Trans. cell	Evidence
5	E(5, 1), E(5, 2), E(5, 3), E(5, 4), E(5, 6), E(5, 7), E(5, 8), E(5, 9)
6	E(6, 2), E(6, 3), E(6, 5), E(6, 8), E(6, 9)
5	E(5, 1), E(5, 2), E(5, 3), E(5, 4), E(5, 6), E(5, 7), E(5, 8), E(5, 9)
Derived	E(5, 6, 2), E(5, 6, 3), E(5, 6, 8), E(5, 6, 9), E(6, 5, 1), E(6, 5, 2), E(6, 5, 3), E(6, 5, 4), E(6, 5, 7), E(6, 5, 8), E(6, 5, 9)

Thus, if the cell dimension is large enough such that the BS's cell contains many neighboring nodes, the ET attack model will fail to reveal the BS cell. On the other hand, to increase the adversary's confidence in determining the BS cell, we should decrease the cell size. Therefore, there is a trade-off between adversary's confidence and complexity by using this attack model. As pointed out earlier, the injection of redundant traffic far from the BS seems to be a remarkable anonymity boosting technique against TV analysis and GSAT test. Simply, nodes that are positioned far from the BS can broadcast redundant packets periodically aiming to increase the transmissions in their regions and subsequently to boost the BS anonymity. In contrast, the ET attack model does not only capture events but also correlate them intending to identify routing paths. The deception of ET attack model requires the injection of redundant traffic into well-defined areas, but the redundant traffic should follow a pattern that is radically different from the normal.

ET tries to locate the BS's cell as an end of most inferred paths (sink). It is not risqué to say that statistical correlation attack is the most powerful and sophisticated attack model, but requires extremely high computational power to apply when assuming a small cell size. Therefore, in order to reduce the complexity of ET analysis without reducing adversary's confidence (by considering bigger cell size), we can reduce the branching factor (to be less than N^2) in Eq. (5.3). If we set the branching factor to one, ET composes paths consisting of one single cell and the complexity becomes linear. That is exactly what TV traffic analysis does, but TV tries to identify the BS's cell as a hot spot area based on a threshold. In general, despite the cell size and the attack model, the BS's cell can be a sink or a hot spot or none if a countermeasure completely changes the traffic pattern.

6 A Novel Traffic Analysis Attack Model and Base Station Anonymity Metrics

In the previous section, we have analyzed attack models that are commonly used to validate the effectiveness of BS anonymity boosting techniques. In general, these attack models can be classified based on the methodology into two categories. Traffic volume (TV) and GSAT test belong to the transmission rate category, and evidence theory (ET) falls under the statistical correlation category. The performance of these

attack models depends on the cell size and the countermeasure technique that is applied to distribute the traffic across the network. Cells are used by the adversary to express uncertainty in accurately locating the source of a signal transmission. Considering a small cell size for the traffic analysis generally spoofs the TV attack model and increases significantly the convergence time for GSAT test; meanwhile, considering a large cell size may nullify the ET attack model. Although ET is the most sophisticated attack model, its convergence time can be an obstacle. The trade-off between ET traffic analysis complexity and adversary's confidence can be balanced by choosing a cell side of $(2/3)r$, where r is the sensor's node transmission range [25]. Then, the complexity can be further reduced by applying ET traffic analysis for shorter time windows as explained in Sect. 5.3. The larger the time window, the higher the complexity and confidence for adversary's traffic analysis.

It is obvious that to further reduce the complexity of ET traffic analysis in Eq. (5.3), we should reduce the number of cells that ET is applied to. Therefore, we need to identify a set M of cells, where $|M| \leq N^2$, and exclude the other $(N^2 - |M|)$ cells from the analysis. It is crucial to carefully select members of M such that the BS's cell is among them. To achieve such an objective, we present a sophisticated attack model, named, *Evidence theory Analysis with Reduced search Space* (EARS), that combines the advantages of the TV and ET attack models. As we pointed out in Sect. 5.1, TV analysis can be an attractive approach because of its low complexity which is linear in the number of cells. For that reason, EARS first uses TV analysis to identify M that consists of traffic hot spot and sink cells and then ET is applied while only considering the cells in M. In other words, TV helps ET in eliminating possible data paths by pointing out a subset of the cells in the grid that should be focused on. In the rest of this section, we describe how EARS overcomes the weaknesses of the existing models and locates the BS more efficiently [25]. Additionally, we present two metrics that can be further used to evaluate a countermeasure.

6.1 EARS's Hot Spot Cells Identification Phase

As we pointed out in the previous section, the cell size affects both TV and ET-based analysis. For ET, the recommendation is to make the cell side equal to $(2/3)r$, where r is the transmission range of the nodes. For TV, as the cell size decreases, the BS's cell may not be a hot spot cell anymore and instead the BS could be located in one of the hot spot neighboring cells. Basically, as the cell size gets smaller, fewer of the BS's 1-hop neighboring nodes will belong to BS's cell and cause it to become a hot spot rather than sink cell. In fact, unlike the case of large cells, a countermeasure can easily create local maxima in the traffic distribution without imposing major redundant traffic when the cell size is small. On the other hand, a small cell size better reveals the routing topology and improves the fidelity of the TV analysis. The fact that the complexity of TV analysis is linear gives us the opportunity to apply it repeatedly for different cell sizes. Hence, EARS applies TV analysis for different grids (number of cells) in order to be able to determine the local maxima that a

countermeasure may create, get a better insight about the traffic distribution across the network, and identify the real hot spots with high probability.

By injecting redundant traffic, a countermeasure usually aims to either balance the traffic density across the network or create hot spot areas far from the BS [13–24]. A major challenge for TV attack model is how to determine the value of threshold β since it depends on the grid size and can be set high for small grid sizes or low for big grid sizes. For that reason, EARS sets threshold β equal to the average p_i of the grid's cells, which equals to $1/N^2$, where N^2 is the number of cells in a $N \times N$ grid. If a cell i in a grid has a probability p_i that exceeds threshold β, cell i is considered to be a hot spot for that grid. A considerably low value of threshold β does not ensure that the BS's cell or a BS's neighboring cell would be identified as a hot spot if a countermeasure is applied. To overcome this issue, EARS initially divides the monitoring area into a 2×2 grid and computes the probability p_i of each cell i ($i = 1, \ldots, 4$), as explained in Sect. 5.1. Then, EARS repeatedly divides the area into a finer-grained grid, i.e., with smaller cells, until the cell side is equal to $2r/3$, and again computes the probability p_i of each grid's cell. We set the minimum cell side to $2r/3$ in order to limit the complexity of evidence theory when applied at a later stage. EARS identifies the hot spots for each of the considered grid sizes based on the associated threshold β, i.e., average p_i for that grid.

Finally, EARS identifies the high-fidelity hot spot regions by considering the traffic distribution in all grids. That is, if a region of the monitoring area is defined as a hot spot in at least half of the considered grids (i.e., by applying simple majority vote among the considered grids), then that region is marked as a high-fidelity hot spot. The rationale for such a technique is to identify the hot spot regions with respect to contiguous traffic flows while excluding hot spot areas that a countermeasure creates. Thus, EARS decides if a cell i from the final grid G_f (with the smallest cells) is a hot spot if and only if the cell's i area is identified as a hot spot in at least half of the considered grids. Basically, the area of cell i in G_f may be covered by a single (bigger) cell j in another grid G_q. In such a scenario, if cell j was identified as a hot spot for G_q, then the area of cell i is implicitly a hot spot. However, when the area of cell i is covered by a set C of multiple cells in G_q, the fraction of cell's i area that is considered hot spot in G_q is calculated based on whether the cells in C are hot spots or not. For example, in Fig. 16, if cells x and y in G_q (grid with continuous lines) are hot spots and cover 39 and 14% of the cell's i area (yellow cell) in grid G_f (grid with dash lines), respectively, then 53% of cell i is considered a hot spot in G_q. Generally, the decision on whether cell i is considered a hot spot w.r.t. G_q depends on whether 50% or more of the cell's i area is covered by hot spot cells in G_q.

To illustrate the space reduction phase through TV analysis considering the sample network in Fig. 17a where no countermeasure is applied, each node transmits periodically 15 packets that are forwarded to the BS over the shortest paths. The blue arrows represent the routing topology, and the annotated numbers indicate the number of packets each node transmits. Figures 17 and 18 demonstrate how EARS reduces space and highlight its advantage over ET. EARS first applies TV using different grid sizes, as illustrated in Fig. 17b–d where the number of transmissions occurring at each cell is shown in the red or the yellow boxes. In a $(N \times N)$ grid, a hot

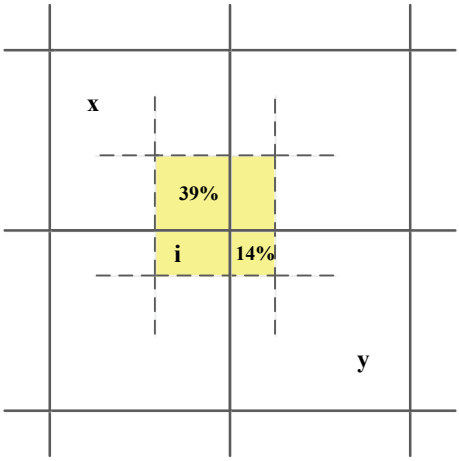


Fig. 16 Identify the area of cell *i*, in the smallest grid that EARS considers, as a hot spot in a bigger grid

spot cell is the one experiencing at least the average number of transmissions ($435/N^2$ in the example in Fig. 17), which constitutes the average probability of hosting the BS according to TV analysis (since each node has the same transmission range and all packets are of the same size). In Fig. 17, hot spot cells are marked with red boxes. The cells with yellow boxes are the cells with probability lower than the average.

EARS then determines the set F of hot spot cells in the biggest grid (4×4 for this example) by using the traffic distributions from all grids (2×2, 3×3, 4×4). If the area of a cell in the 4×4 grid is a hot spot in at least one more grid (i.e., total of two out of three grids), then such a cell is considered to be a hot spot and is placed in set F. Figure 18 shows the cells that are identified as hot spots (set F) with a red box within them. It is easy to map the cells from 4×4 to 2×2 grid and decide which cells are hot spots in both grids (cells 10 and 14). On the other hand, a cell j in the 4×4 grid may be covered by 1 or 2 or 4 cells in the 3×3 grid. EARS handles that by considering all the cells in smaller grids that cover a cell j in a bigger grid. For example, the area of cell 11 in the 4×4 grid is covered by 4 cells in the 3×3 grid. Cells 5, 6, 8, and 9 in the 3×3 grid cover 45%, 22%, 22%, and 11% of cells' 11 area, respectively. Cells 5, 8, and 9 in the 3×3 grid are identified as hot spots, and hence, 78% (>50%) of the area of cell is identified as hot spot in the 3×3 grid counterpart. Therefore, cell 11 is identified as a hot spot for 2 out of 3 grids (3×3 grid and 4×4 grid) and is added to set F. As we can see in Fig. 18, EARS identifies hot spot cells with higher confidence while increasing the complexity only by a linear factor as we show later.

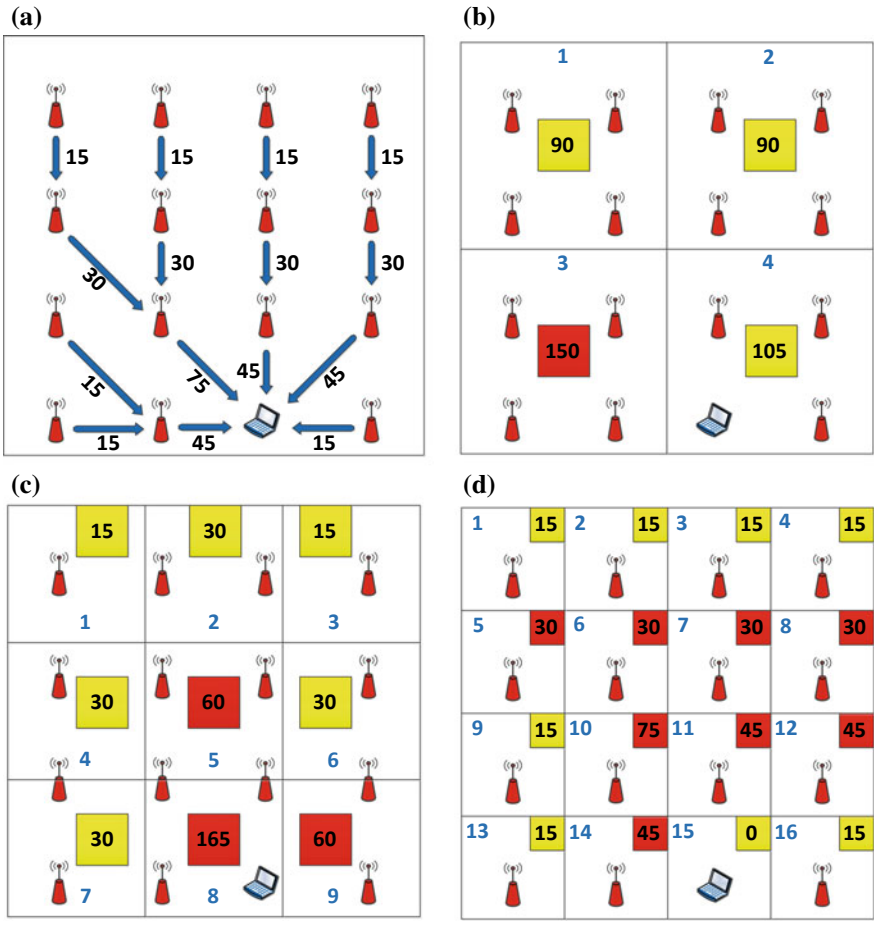


Fig. 17 TV analysis applied for different grid sizes

6.2 EARS's Sink Cell Identification Phase

To overcome the shortcoming of TV analysis where the BS's cell could be a sink instead of a hot spot, i.e., the BS's cell is a neighbor of hot spot cells, EARS expands the set of considered cells. The idea is to augment the set "F" of hot spots with the candidate sink cells to form the set "M". Although it can simply be assumed that every cell that is a neighbor of a hot spot may be a sink and may be added to set "F", the number of cells will grow substantially and the advantage of EARS diminishes. Instead, EARS qualifies the neighbors (cells) of hot spot cells in order to determine the most relevant sink cells. Basically, neighbors (cells) of the hot spots with the highest probabilities of having the BS based on TV form the set S. First, EARS determines a set $D \subseteq F$, consisting of the high-fidelity hot spots among the cells in

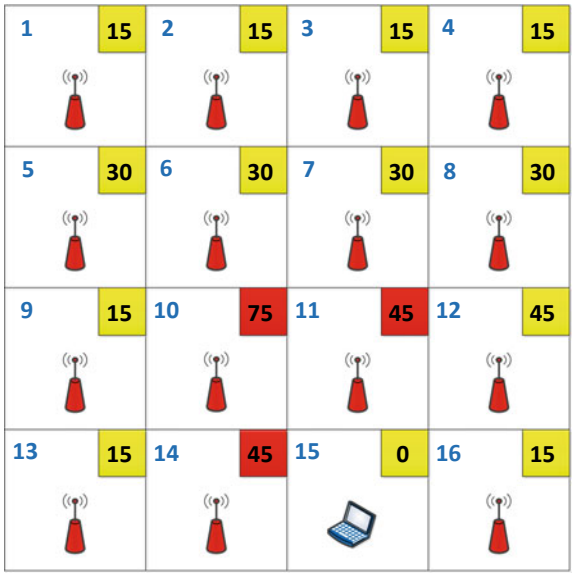


Fig. 18 EARS identifies set F (hot spot cells)

F, i.e., those with the highest probabilities. To do so, EARS identifies the cells of F that have p_i higher than the average probability over all members of the set F. These cells are the members of the set D. Finally, EARS defines a set S of highly probable sink cells which are adjacent to any member of D, i.e., high-fidelity hot spots in F. Thus, the set of cells M, where $|M| \leq N^2$, considered by EARS for applying ET traffic analysis equals $S \cup F$.

How EARS accounts for sink cells is demonstrated in Fig. 19, based on the hot spots in Fig. 18. Only one cell is identified as a high-fidelity hot spot (set D), that is, the cell with the purple box (cell 10) while all the adjacent cells (with the orange boxes) are identified as high probable sinks. Thus, the set M consists of cells with red (hot spots), purple (high-fidelity hot spot), and orange (highly probable sinks) boxes. It is worth noting that a cell that appears as a hot spot in the 4×4 grid, e.g., cell 8 is excluded from M. This highlights the importance of voting across grids, which fundamentally masks cells with high traffic volume due to packet relaying rather than the presence of the BS. Additionally, cell 15 that hosts the BS was identified as a hot spot only in the 3×3 grid (one out of three), but EARS succeeds to identify it as a sink cell and include it in set M. This shows that EARS overcomes the disadvantage of TV analysis which assumes that the BS's cell is a hot spot.

ET traffic analysis can now be applied to the cells of M, which collectively constitutes only 56% of the total monitoring area. Additionally, instead of correlating 435 packet transmissions to derive all possible paths for 4×4 grid, we correlate 285 (65%) packet transmissions for a 3×3 grid. In other words, EARS not only reduces the complexity of the ET analysis by decreasing the search space, but also by excludes

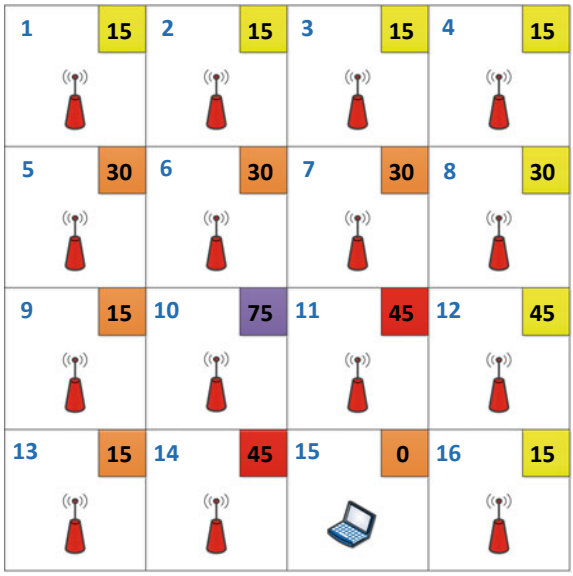


Fig. 19 EARS's reduced space M (cells with purple, red, and orange boxes)

a significant portion of collected evidences (transmissions) that ET analysis would have correlated.

6.3 *EARS Complexity Analysis and Anonymity Metrics*

EARS uses TV analysis repeatedly to capture the hot spot regions based on traffic flows in order to filter out local maxima that a countermeasure may have been introduced. After hot spot regions are identified with high confidence, EARS identifies the high probable sinks to capture the BS's cell as a sink. As shown in [25], the complexity of the EARS space reduction process is $O\left(k \times N^2\right)$, where k is the number of different grids EARS uses and N^2 is the number of cells in the final grid, i.e., with the smallest cells. Finally, EARS applies ET traffic analysis in the reduced space. Assuming that EARS reduces the search space for ET analysis by a factor $\gamma\%$, then runtime complexity of EARS is as follows:

$$Complexity = (1-\gamma)N^2 \times O\left(\left(\sum_{i=1}^{q} 8 \times i\right)^{(1-\gamma)N^2}\right) + O\left(kN^2\right), \qquad (6.1)$$

The first term is the complexity that ET analysis requires, while the second term is what EARS takes to reduce the space (using TV analysis). The second term is dominated by the exponential cost of ET, even with the reduced space. Figure 20 shows the complexity reduction with respect to different grid sizes that EARS provides by

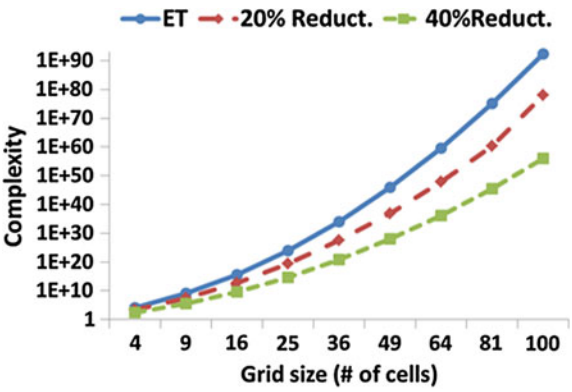


Fig. 20 Illustration of the effect of EARS on reducing the runtime complexity of the traffic analysis

reducing the space, i.e., 20 and 40% of the total number of cells in the grid, compared to ET analysis that considers the whole grid. Figure 20 shows how reduction becomes more significant when the number of cells increases and demonstrates that EARS enables fine-grained analysis of the traffic.

An additional advantage of EARS is the introduction of two anonymity metrics that can be used to evaluate the performance of countermeasures. The first metric is based on the threshold β and assesses how much a countermeasure alters the traffic pattern in the network. A low value of β ensures that the set M includes the BS's cell even when a powerful countermeasure is applied to the network. The lower the value of threshold β should be set, the better a countermeasure alters the traffic pattern, meaning that a countermeasure creates high local maxima in the traffic distribution such that we should set threshold β low to capture the BS's cell or BS's vicinity as a hot spot. The second metric is the size of the set M and reveals how well a countermeasure distributes the traffic across the network. The higher the $|M|$ is, the better the distribution of the redundant traffic across the network.

6.4 EARS Evaluation

EARS reduces the scale of the analysis by excluding parts of the network that are not identified as hot spots or possible sinks of the collected traffic. Thus, EARS enables finer analysis of the traffic distribution which results in higher adversary's confidence in localizing the BS. The effectiveness of EARS is validated through simulation using sample countermeasures. EARS performance has been evaluated under both network operation models, (i) event-triggered, where packets are generated when sensors detect something noteworthy to report on, and (ii) periodic, where sensors send samples every predefined time period. In addition, we show how effective the proposed anonymity metrics, namely size of set M and threshold β, are for countermeasure's evaluation. In this subsection, we show a subset of the results

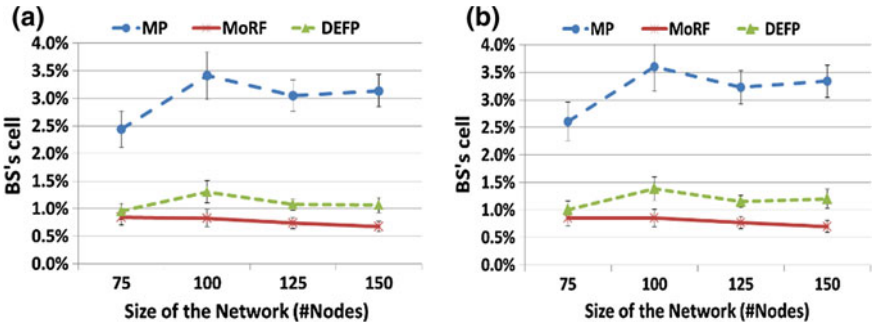


Fig. 21 Impact of network size on the normalized weighted Belief value of the BS's cell that ET computes (**a**) and that EARS computes (**b**)

while only considering the periodic data sampling experiments. More results can be found in [25].

The simulation experiments involve 75–150 sensor nodes whose communication range is set to 150 m and are deployed in 1200×1200 m^2 area. As we discussed in Sect. 6.1, in EARS, the adversary divides the monitoring area into different grid sizes. The minimum cell size that EARS reaches is $(2/3) \times 150\,\text{m} = 100\,\text{m}$, making the biggest grid (12×12). Hence, for EARS, we divide the monitoring area into grids of sizes $(2 \times 2), (3 \times 3), (4 \times 4), \ldots, (12 \times 12)$. EARS computes for each countermeasure the set M, as described earlier. We set the threshold β equal to the average p_i for the cells of the grid under consideration. To validate that EARS successfully locates the BS, we applied ET analysis into the whole space of the 12×12 grid and we computed the weighted Belief value of the BS's cell for MP [4], DEFP [4], and MoRF [23] countermeasures. The extremely high number of collected evidences in periodic networks necessitates the use of short time windows. We use time windows of 7 s in the composed derived paths in order to reduce the complexity that the 12×12 grid imposes to all attack models.

Figure 21 shows the normalized weighted Belief value of the BS's cell for different network sizes, when ET is applied to the whole monitoring area (144 cells) (Fig. 21a) and when ET is applied into the reduced space (set M) generated by EARS (Fig. 21b). The results are averaged over 25 simulation runs. The error bars represent the standard error of each measurement. As we can see, EARS matches and often slightly exceeds ET for all countermeasures. Additionally, it is notable to report that the monotonicity of the normalized weighted Belief value function remains the same as the network size increases when EARS is applied.

Finally, we show the importance of the threshold β to the performance of EARS. In Fig. 22, we vary threshold β and report the size of the set M for all countermeasures. Figure 22 expresses the adversary's uncertainty for all countermeasures as a ratio of the number of times BS's cell is not included in M divided by the number of simulation runs. The results are averaged over 50 random-shaped topologies of 150 nodes. The error bars represent the standard error of each measurement.

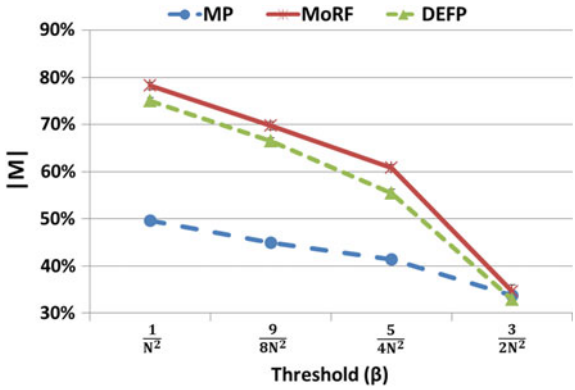


Fig. 22 Effect of the threshold β on EARS's reduced space

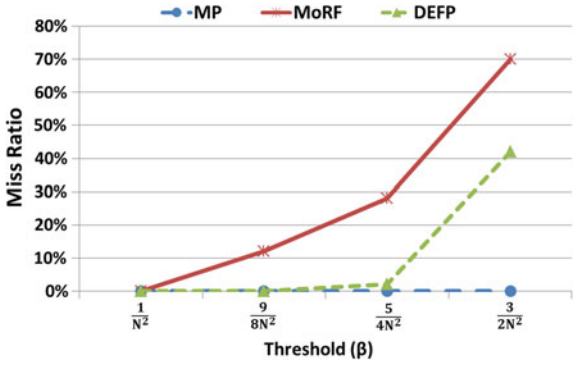


Fig. 23 Effect of the threshold β on the miss ratio for each countermeasure

Figure 22 indicates that as the threshold β increases, the size of M decreases. MoRF has the largest $|M|$ for all values of β. While it seems that we can decrease significantly the complexity of ET analysis as the threshold increases, we cannot ensure the BS's cell is in set M for both countermeasures. In Fig. 23, we see how many times EARS has failed to identify the BS's cell as a hot spot or sink for the various values of β. We express the uncertainty of having the BS's cell in M, for a given threshold β as a percentage of misses. That percentage indicates what threshold is appropriate for each countermeasure. When a countermeasure has low miss ratio for high threshold β, it means that the countermeasure does not alter the traffic distribution significantly.

From the simulation results, we can conclude that EARS successfully locates the BS with at least the same accuracy of ET analysis when applied to the entire area. EARS's complexity reduction enables the use of bigger grids and time windows and hence further reduces adversary's uncertainty. EARS also allows to mute the effect of countermeasures that just create traffic hot spots throughout the network and thus helps the application designer gauge the resiliency of the network to traffic analysis attacks and uncover vulnerabilities. Furthermore, the two anonymity metrics, namely size of set M and threshold β, that EARS introduces are shown to be invaluable in

evaluating how much a countermeasure alters the traffic pattern and how well it distributes the overhead across the network.

7 Conclusion

In this chapter, we have discussed different traffic analysis attack models and summarized published countermeasures that opt to preserve location privacy in WSN. Traffic analysis is a major threat to WSN, where an adversary may eavesdrop on transmissions and correlate them in order to uncover the location of a target entity. Source location privacy has attracted significant attention from the research community since the scope of many applications is to protect the reported event. On the other hand, sustaining the BS's location privacy is the most important given the crucial role that the BS plays. In general, it is easy to provide single-source anonymity in an event-triggered network since energy and performance constraints are relaxed. In contrast, hiding the BS's vicinity from a sophisticated adversary while continuous traffic flows toward and merges in the vicinity of the BS requires countermeasures that significantly impact the network performance and lifetime.

Published location privacy-preserving techniques have mainly assumed three models for how the traffic analysis can be performed. In this chapter, we have analyzed these models and pointed out shortcomings that may lead to wrongfully assume degraded adversary's capabilities than in reality. TV analysis and GSAT test have low complexity, but the injection of redundant traffic far from the BS seems to be a remarkable anonymity boosting technique. Simply, nodes that are positioned far from the BS can broadcast redundant packets periodically aiming to increase the transmissions in their regions and subsequently to boost the BS anonymity. In contrast, the ET attack model not only captures events but also correlates them intending to identify routing paths. Injection of redundant traffic into well-defined areas does not suffice for the deception of ET attack model, and the redundant traffic should follow a sophisticated traffic pattern.

Additionally, we have presented EARS as a novel traffic analysis attack model that addresses the shortcomings of the existing attack models and better reflects what an adversary could do. EARS introduces two new anonymity assessment metrics that can be further used to gauge the effectiveness of anti-traffic analysis techniques. EARS raises the bar for the countermeasures and motivates the need for a fresh look on how the network asset is protected in practice and for developing more robust techniques for preserving the location privacy of the BS. Finally, as an open research problem, we would like to stress the importance of providing multi-source and sink location privacy, simultaneously, with respect to the network's performance and resources. This type of location privacy-preserving problem necessitates the design of anti-traffic analysis techniques that are nondeterministic and aim to attract adversary's attention on different network areas over time based on the various events' locations. Achieving such a goal while minimizing the incurred overhead is quite challenging.

References

1. 21 ideas for the 21st century. Business Week, pp. 78–167 (1999)
2. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Comput. Netw. **38**(4), 393–422 (2002)
3. Chong, C.Y., Kumar, S.P.: Sensor networks: evolution, opportunities, and challenges. Proc. IEEE **91**(8), 1247–1256 (2003)
4. Deng, J., Han, R., Mishra, S.: Countermeasures against traffic analysis attacks in wireless sensor networks. In: First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05), pp. 113–126, Sept 2005. IEEE
5. Kong, J., Xiaoyan, H., Gerla, M.: An identity-free and on-demand routing scheme against anonymity threats in mobile ad hoc networks. IEEE Trans. Mob. Comput. **6**(8), 888–902 (2007)
6. Venkitasubramaniam, P., He, T., Tong, L., Wicker, S.B.: Toward an analytical approach to anonymous wireless networking. IEEE Commun. Mag. **46**(2), 140–146 (2008)
7. Qin, Y., Huang, D., Li, B.: STARS: a statistical traffic pattern discovery system for MANETs. IEEE Trans. Dependable Secure Comput. **11**(2), 181–192 (2014)
8. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.E.: SPINS: security protocols for sensor networks. Wirel. Netw. **8**(5), 521–534 (2002)
9. Conner, W., Abdelzaher, T., Nahrstedt, K.: Using data aggregation to prevent traffic analysis in wireless sensor networks. In: International Conference on Distributed Computing in Sensor Systems, pp. 202–217, June 2006. Springer Berlin Heidelberg
10. Ebrahimi, Y., Younis, M.: Averting in-situ adversaries in wireless sensor network using deceptive traffic. In: Global Telecommunications Conference (GLOBECOM 2011), pp. 1–5, Dec 2011. IEEE
11. Li, X., Wang, X., Zheng, N., Wan, Z., Gu, M.: Enhanced location privacy protection of base station in wireless sensor networks. In: 5th International Conference on Mobile Ad-hoc and Sensor Networks (MSN'09), pp. 457–464, Dec 2009. IEEE
12. Li, N., Zhang, N., Das, S.K., Thuraisingham, B.: Privacy preservation in wireless sensor networks: a state-of-the-art survey. Ad Hoc Netw. **7**(8), 1501–1514 (2009)
13. Acharya, U., Younis, M.: Increasing base-station anonymity in wireless sensor networks. Ad Hoc Netw. **8**(8), 791–809 (2010)
14. Bicakci, K., Bagci, I.E., Tavli, B.: Lifetime bounds of wireless sensor networks preserving perfect sink unobservability. IEEE Commun. Lett. **15**(2), 205–207 (2011)
15. Ebrahimi, Y., Younis, M.: Increasing transmission power for higher base-station anonymity in wireless sensor network. In: 2011 IEEE International Conference on Communications (ICC), pp. 1–5, June 2011. IEEE
16. Ebrahimi, Y., Younis, M.: Using deceptive packets to increase base-station anonymity in wireless sensor network. In: 2011 7th International Wireless Communications and Mobile Computing Conference, pp. 842–847, July 2011. IEEE
17. Ying, B., Gallardo, J.R., Makrakis, D., Mouftah, H.T.: Concealing of the sink location in WSNs by artificially homogenizing traffic intensity. In: 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 988–993, Apr 2011. IEEE
18. Mehta, K., Liu, D., Wright, M.: Location privacy in sensor networks against a global eavesdropper. In: 2007 IEEE International Conference on Network Protocols, pp. 314–323, Oct 2007. IEEE
19. El-Badry, R., Younis, M.: Providing location anonymity in a multi-base station wireless sensor network. In: ICC, pp. 157–161, June 2012
20. Ward, J.R., Younis, M.: On the use of distributed relays to increase base station anonymity in wireless sensor networks. In: MILCOM 2012-2012 IEEE Military Communications Conference, pp. 1–6. IEEE
21. Ward, J.R., Younis, M.: On the use of distributed beamforming to increase base station anonymity in wireless sensor networks. In: 2013 22nd International Conference on Computer Communication and Networks (ICCCN), pp. 1–7, July 2013. IEEE

22. Di Ying, B., Makrakis, D., Mouftah, H.T.: Anti-traffic analysis attack for location privacy in WSNs. EURASIP J. Wirel. Commun. Netw. **2014**(1), 1 (2014)
23. Baroutis, N., Younis, M.: Using fake sinks and deceptive relays to boost base-station anonymity in wireless sensor network. In: 2015 IEEE 40th Conference on Local Computer Networks (LCN), pp. 109–116, Oct 2015. IEEE
24. Baroutis, N., Younis, M.: Boosting base-station anonymity in wireless sensor networks through illusive multiple-sink traffic. In: 2016 IEEE Global Communications Conference (GLOBECOM 2016), Dec 2016. IEEE
25. Baroutis, N., Younis, M.: A Novel Traffic Analysis Attack Model and Base-station Anonymity Metrics for Wireless Sensor Networks. Security and Communication Networks (to appear)
26. Alsemairi, S., Younis, M.: Forming a cluster-mesh topology to boost base-station anonymity in wireless sensor networks. In: Wireless Communications and Networking Conference (WCNC), pp. 1–6, Apr 2016. IEEE
27. Alsemairi, S., Younis, M.: Clustering-based mitigation of anonymity attacks in wireless sensor networks. In: 2015 IEEE Global Communications Conference (GLOBECOM), pp. 1–7, Dec 2015. IEEE
28. Alsemairi, S., Younis, M.: Adaptive packet-combining to counter traffic analysis in wireless sensor networks. In: 2015 International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 337–342, Aug 2015. IEEE
29. Gottumukkala, V.P.V., Pandit, V., Li, H., Agrawal, D.P.: Base-station location anonymity and security technique (BLAST) for wireless sensor networks. In: 2012 IEEE International Conference on Communications (ICC), pp. 6705–6709, June 2012. IEEE
30. Bangash, Y., Zeng, L., Feng, D.: MimiBS: mimicking base-station to provide location privacy protection in wireless sensor networks. In: 2015 IEEE International Conference on Networking, Architecture and Storage (NAS), pp. 158–166, Aug 2015. IEEE
31. Ozturk, C., Zhang, Y., Trappe, W.: Source-location privacy in energy-constrained sensor network routing. In: Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks, pp. 88–93, Oct 2004. ACM
32. Kamat, P., Zhang, Y., Trappe, W., Ozturk, C.: Enhancing source-location privacy in sensor network routing. In: 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05), pp. 599–608, June 2005. IEEE
33. Xi, Y., Schwiebert, L., Shi, W.: Preserving source location privacy in monitoring-based wireless sensor networks. In: Proceedings 20th IEEE International Parallel & Distributed Processing Symposium, pp. 8, Apr 2006. IEEE
34. Mehta, K., Liu, D., Wright, M.: Protecting location privacy in sensor networks against a global eavesdropper. IEEE Trans. Mob. Comput. **11**(2), 320–336 (2012)
35. Li, Y., Ren, J.: Preserving source-location privacy in wireless sensor networks. In: 2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, pp. 1–9, June 2009. IEEE
36. Chen, H., Lou, W.: From nowhere to somewhere: protecting end-to-end location privacy in wireless sensor networks. In: International Performance Computing and Communications Conference, pp. 1–8, Dec 2010. IEEE
37. Spachos, P., Song, L., Bui, F.M., Hatzinakos, D.: Improving source-location privacy through opportunistic routing in wireless sensor networks. In: 2011 IEEE Symposium on Computers and Communications (ISCC), pp. 815–820, June 2011. IEEE
38. Lightfoot, L., Li, Y., Ren, J.: Preserving source-location privacy in wireless sensor network using STaR routing. In: Global Telecommunications Conference (GLOBECOM 2010), pp. 1–5, Dec 2010. IEEE
39. Gurjar, A., Patil, A.B.: Cluster based anonymization for source location privacy in wireless sensor network. In: 2013 International Conference on Communication Systems and Network Technologies (CSNT), pp. 248–251, Apr 2013. IEEE
40. Li, Y., Lightfoot, L., Ren, J.: Routing-based source-location privacy protection in wireless sensor networks. In: 2009 IEEE International Conference on Electro/Information Technology, pp. 29–34, June 2009. IEEE

41. Wang, H., Sheng, B., Li, Q.: Privacy-aware routing in sensor networks. Comput. Netw. **53**(9), 1512–1529 (2009)
42. Ouyang, Y., Le, Z., Chen, G., Ford, J., Makedon, F.: Entrapping adversaries for source protection in sensor networks. In: Proceedings of the 2006 International Symposium on World of Wireless, Mobile and Multimedia Networks, pp. 23–34, June 2006. IEEE Computer Society
43. Kazatzopoulos, L., Delakouridis, C., Marias, G.F., Georgiadis, P.: ihide: Hiding sources of information in wsns. In: Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU'06), pp. 8, June 2006. IEEE
44. Yang, Y., Shao, M., Zhu, S., Urgaonkar, B., Cao, G.: Towards event source unobservability with minimum network traffic in sensor networks. In: Proceedings of the First ACM Conference on Wireless Network Security, pp. 77–88, Mar 2008. ACM
45. Alomair, B., Clark, A., Cuellar, J., Poovendran, R.: Toward a statistical framework for source anonymity in sensor networks. IEEE Trans. Mob. Comput. **12**(2), 248–260 (2013)
46. Mahmoud, M.M., Shen, X.: A cloud-based scheme for protecting source-location privacy against hotspot-locating attack in wireless sensor networks. IEEE Trans. Parallel Distrib. Syst. **23**(10), 1805–1818 (2012)
47. Bicakci, K., Gultekin, H., Tavli, B., Bagci, I.E.: Maximizing lifetime of event-unobservable wireless sensor networks. Comput. Stand. Interfaces **33**(4), 401–410 (2011)
48. Yang, W., Zhu, W.T.: Protecting source location privacy in wireless sensor networks with data aggregation. In: International Conference on Ubiquitous Intelligence and Computing, pp. 252–266, Oct 2010. Springer Berlin Heidelberg
49. Misra, S., Xue, G.: Efficient anonymity schemes for clustered wireless sensor networks. Int. J. Sensor Netw. **1**(1–2), 50–63 (2006)
50. Zhang, L., Zhang, H., Conti, M., Di Pietro, R., Jajodia, S., Mancini, L.V.: Preserving privacy against external and internal threats in WSN data aggregation. Telecommun. Syst. **52**(4), 2163–2176 (2013)
51. Rahman, F., Hoque, E., Ahamed, S.I.: Preserving privacy in wireless sensor networks using reliable data aggregation. ACM SIGAPP Appl. Comput. Rev. **11**(3), 52–62 (2011)
52. Ngai, E.C.H., Rodhe, I.: On providing location privacy for mobile sinks in wireless sensor networks. Wirel. Netw. **19**(1), 115–130 (2013)
53. Oh, S., Gruteser, M.: Multi-node coordinated jamming for location privacy protection. In: 2011-MILCOM 2011 Military Communications Conference, pp. 1243–1249, Nov 2011. IEEE
54. Tavli, B., Ozciloglu, M.M., Bicakci, K.: Mitigation of compromising privacy by transmission range control in wireless sensor networks. IEEE Commun. Lett. **14**(12), 1104–1106 (2010)
55. Shao, M., Hu, W., Zhu, S., Cao, G., Krishnamurth, S., La Porta, T.: Cross-layer enhanced source location privacy in sensor networks. In: 2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, pp. 1–9, June 2009. IEEE
56. Shao, M., Yang, Y., Zhu, S., Cao, G.: Towards statistically strong source anonymity for sensor networks. In: INFOCOM 2008. The 27th Conference on Computer Communications. IEEE
57. Pfitzmann, A., Hansen, M.: Anonymity, Unlinkability, Unobservability, Pseudonymity, and Identity Management-A Consolidated Proposal for Terminology (2005)
58. Seys, S., Preneel, B.: ARM: anonymous routing protocol for mobile ad hoc networks. Int. J. Wirel. Mob. Comput. **3**(3), 145–155 (2009)
59. Zhang, Y., Liu, W., Lou, W.: Anonymous communications in mobile ad hoc networks. In: Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, pp. 1940–1951, Mar 2005. IEEE
60. Boukerche, A., El-Khatib, K., Xu, L., Korba, L.: A novel solution for achieving anonymity in wireless ad hoc networks. In: Proceedings of the 1st ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, pp. 30–38, Oct 2004. ACM
61. Conti, M., Willemsen, J., Crispo, B.: Providing source location privacy in wireless sensor networks: a survey. IEEE Commun. Surv. Tutor. **15**(3), 1238–1280 (2013)

62. Park, H., Song, S., Choi, B.Y., Huang, C.T.: Passages: preserving anonymity of sources and sinks against global eavesdroppers. In: INFOCOM, 2013 Proceedings IEEE, pp. 210–214, Apr 2013. IEEE
63. Li, X., Shi, H., Shang, Y.: A sorted RSSI quantization based algorithm for sensor network localization. In: 11th International Conference on Parallel and Distributed Systems (ICPADS'05), vol. 1, pp. 557–563, July 2005. IEEE
64. Yang, Y., Lee, J., Jung, J., Song, S., Yoon, H., Yoon, Y.: Target source detection using an improved sensing model in wireless sensor networks (ISMWSNs). In: 2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 5899–5902, Aug 2007. IEEE
65. Savvides, A., Han, C.C., Strivastava, M.B.: Dynamic fine-grained localization in ad-hoc networks of sensors. In: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, pp. 166–179, July 2001. ACM
66. Elnahrawy, E., Li, X., Martin, R.P.: The limits of localization using signal strength: a comparative study. In: 2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, IEEE SECON 2004, pp. 406–414. IEEE
67. Bensky, A.: Wireless Positioning Technologies and Applications. Artech House (2016)
68. Chandrasekaran, G., Ergin, M.A., Yang, J., Liu, S., Chen, Y., Gruteser, M., Martin, R.P.: Empirical evaluation of the limits on localization using signal strength. In: 2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, pp. 1–9, June 2009. IEEE
69. Selman, B., Levesque, H.J., Mitchell, D.G.: A new method for solving hard satisfiability problems. In: AAAI, vol. 92, pp. 440–446, July 1992
70. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27**, 379–423, 623–656 (1948)
71. Diaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: International Workshop on Privacy Enhancing Technologies, pp. 54–68, Apr 2002. Springer Berlin Heidelberg
72. Serjantov, A., Danezis, G.: Towards an information theoretic metric for anonymity. In: International Workshop on Privacy Enhancing Technologies, pp. 41–53, Apr 2002. Springer Berlin Heidelberg
73. Huang, D.: On measuring anonymity for wireless mobile ad-hoc networks. In: Proceedings 2006 31st IEEE Conference on Local Computer Networks, pp. 779–786, Nov 2006. IEEE

Implementation of Secure Communications for Tactical Wireless Sensor Networks

Preetha Thulasiraman and David W. Courtney

Abstract The ability to securely disseminate data in a timely manner is critical to military missions within a hostile environment. Tactical wireless sensor networks (WSN) consist of power-constrained devices spread throughout a region-of-interest to provide data extraction in real time. In this chapter, we discuss cyber security mechanisms to be implemented on a tactical WSN using the 6LoWPAN protocol for use by the United States Marine Corps (USMC). Specifically, we develop an architectural framework for tactical WSNs by studying cyber security gaps and vulnerabilities within the 6LoWPAN security sublayer, which is based on the IEEE 802.15.4 standard. We present a key management scheme and a centralized routing mechanism that is non-broadcast but feasible in an operational scenario. In addition, we modify the 6LoWPAN enabled IEEE 802.15.4 frame structure to facilitate the newly developed keying and centralized routing mechanisms. Methods to aid in deployment planning are also discussed. The tactical WSN architecture was tested against a variety of well-known network attacks. The attacks simulated were spoofing, man-in-the-middle, and denial-of-service. Through MATLAB simulations, we showed the effectiveness and efficiency of the developed cyber security mechanisms to provide integrity and reliability to a deployed tactical WSN.

1 Introduction

The Internet of Things (IoT) embraces network connectivity of everyday, low power sensor devices for two-way data communication. The IoT offers the potential to extend connectivity to sensor devices and mobile nodes at the tactical edge of the battlefield at a low cost. A tactical WSN has different priorities from a WSN used

P. Thulasiraman (✉)
Naval Postgraduate School, Monterey, CA, USA
e-mail: pthulas1@nps.edu

D. W. Courtney
Navy Cyber Warfare Development Group, Suitland, MD, USA
e-mail: david.courtney@navy.mil

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_19

for civilian applications. A tactical WSN is generally deployed in hostile or austere environments where the intentional compromise of nodes is quite high. Specifically, a tactical WSN requires situational awareness of its environment, more so than a WSN that is used commercially. In this chapter, we discuss the security of specific IoT protocols such that they can be used at the tactical edge by the United States Marine Corps (USMC) within their wireless sensor networks (WSN).

1.1 Low Power Wireless Sensor Networks

A WSN is a group of sensor nodes that are geographically distributed to provide data gathering and monitoring of tasks and events [22]. Sensor devices collect and transmit data to a centralized controller, also known as a base station. WSNs can be used in a variety of industrial, commercial and military applications. In recent years, WSNs have become integral to all parts of life and continue to evolve as technology advances. The use of WSNs has continuously grown and is becoming fully integrated within commercial systems as well as our homes. Due to the growth of WSNs and the increased reliability of the devices used, a WSN is an attractive intelligence gathering system to the Department of Defense (DOD). The implementation of a WSN provides the ability to remotely collect intelligence on an area-of-interest, eliminating hazardous manpower requirements. In addition, a WSN can be used to remotely monitor deployed systems and trigger alerts at a command and control site when certain events occur [22]. The implementation of tactical WSNs empowers the DOD to cost effectively collect reconnaissance data within critical areas-of-interest, increasing the DOD's intelligence gathering capabilities.

The deployment of tactical WSNs in remote regions requires that the sensor nodes function independently. In order to function for a period of time without manual support, the sensor devices must have their own power supply as a permanent electrical infrastructure might not be readily available. To meet the low energy consumption requirement that extends the life expectancy of the device and, thereby, the network as a whole, WSNs have been designed to use a minimal amount of power to perform a desired task.

The WSN devices in use today by the USMC for tactical networking are known as the AN/GSQ-257, Unattended Ground Sensor Set. The AN/GSQ-257 devices are part of the USMCs Tactical Remote Sensor System and have multiple configurations that enable sensing of seismic/acoustic, magnetic, and/or infrared data [2]. This is helpful in performing perimeter enemy detection and tracking enemy movements. The use of a WSN allows the USMC to remove the human element from possible danger while maintaining situational awareness with early detection from a remote location; however, the limitations of these sensor devices include (1) physical attributes, the weight and size of these sensors are considerably higher than commercial sensor nodes, and (2) limited ability to airdrop, sensors with seismic sensing capability may be air dropped whereas all other sensing functions are only available if the sensors are hand placed. Embracing the technological advancements of sensor and sensor

communication over the past 20 years will enable the USMC to modernize their network infrastructure.

1.2 Introduction to 6LoWPAN/IEEE 802.15.4

One of the advantages of a sensor device is its ability to be deployed within harsh environmental conditions without a dedicated power supply. Since sensors are traditionally low power devices, a different type of communication protocol that conserves energy within the wireless environment needs to be used. The IEEE 802.15.4 standard is a physical and data link communication protocol for low power wireless personal area networks (LoWPAN). LoWPAN is commonly used in embedded applications for real-time data extraction covering a large geographic area requiring the use of many sensor nodes [10]. The sensor nodes within the LoWPAN need to be low cost as well as be able to operate unattended, use typical batteries, and communicate over multiple hops [10].

Since the IEEE 802.15.4 standard only defines the first two layers of the Open Systems Interconnection model, another protocol must be used to provide full networking functionality for the WSN [8]. The Internet Engineering Task Forces (IETF) 6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks) is a protocol designed to work with the IEEE 802.15.4 standard [8]. Both IEEE 802.15.4 and 6LoWPAN are specifically tailored for IoT applications [8]. 6LoWPAN is an open standard networking technology that standardizes Internet connectivity for low power wireless sensor networks. It alters the landscape by allowing IPv6 packets to be carried efficiently within link layer frames, such as those defined by IEEE 802.15.4 [15] while reducing Internet Protocol (IP) overhead. This low overhead is achieved using cross-layer optimizations. A powerful feature of 6LoWPAN is that while originally conceived to support IEEE 802.15.4 low power wireless networks, it is now being adapted and used over a variety of networking media including Bluetooth and Wi-Fi [15]. By connecting to an IP based infrastructure, low power wireless networks are then connected to other IP networks and devices via IP routers. 6LoWPAN takes advantage of the well-developed end-to-end IP infrastructure, providing open standards and interoperability [15].

1.3 Chapter Motivations

As the use of WSNs grow in the USMC, they will become more attractive to potential attackers. In order to prevent a passive or active cyber attack, multiple security methods must be implemented to maintain an efficient and effective WSN. Comprehensive defense security mechanisms must account for multiple types of attacks. Generally, to defend against an attack, the military develops a defense model for the attack. Since there are multiple types of attacks, the military has developed multiple

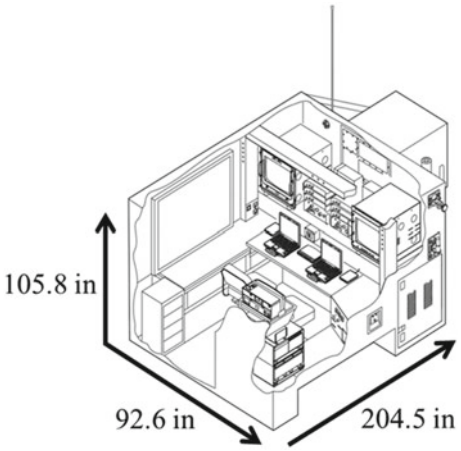


Fig. 1 Dimensions of the COC (AN/MSC-77) currently used by the USMC

models to defend against each case. The development of a single model to defend against a variety of attacks prevents the need for an expanded arsenal of defense models, saving the military manpower.

The USMC has high interest in WSNs and their ability to connect to a public domain. Currently, when their WSN devices are deployed in the field, they are deployed with a base station, known as AN/MSC-77, which contains working spaces for two military personnel [1]. The AN/MSC-77 is also known as the Combat Operations Center (COC). The COC includes a dedicated power source to provide the power necessary to run all of the equipment within it. The COC must be placed in the vicinity of the WSN devices unless a repeater is used to place the unit further away, but the COC must again retain a line-of-sight with the repeater. The size of the COC is a concern since it is a large unit, as shown in Fig. 1, and can be easily seen by an enemy. The current WSN used by the USMC requires the COC to be located near the deployed network even with the use of repeaters; thus, an enemy can generally avoid the area to evade detection. The current data flow from legacy equipment and sensor devices lacks automation. In order for the USMC to obtain the data from the WSN, an individual must physically go to the COC and extract the necessary information, as the COC does not transmit the data acquired from the WSN.

To facilitate seamless data delivery to and from the sensor devices, the network must be connected to another secure domain using a comprehensive communication protocol. The use of 6LoWPAN will significantly improve the information flow as it currently exists by allowing multiple operators in a unit to access sensor information despite their location. IP based information can be easily used to inform the situational awareness and common operational picture of the engaged unit.

The feasibility of using 6LoWPAN for the operational information flow scenario described above is based on its ability to do two things: (1) provide for secure energy conserving communications with regard to the limited power devices that operate over the network and (2) provide secure communications by improving upon the

security sublayer currently utilized by IEEE 802.15.4. The ability to communicate using 6LoWPAN is meaningful only if physical and cyber security considerations are in place.

Our study into the applicability of a 6LoWPAN enabled IEEE 802.15.4 infrastructure for USMC tactical sensor networking is focused on a structured, multi-hop, static WSN rather than an ad hoc deployment. To this end, this chapter aims to provide an analysis of the 6LoWPAN standard by studying the standard's ability to (1) provide energy conservation for the low power devices that are deployed, where the sensors are static and (2) implement and use a key management scheme that has the ability to defend against a variety of cyber security attacks that could cause the WSNs to become inefficient and/or ineffective. In order to achieve this objective, the vulnerabilities of the cyber security mechanisms used within 6LoWPAN WSN is evaluated. A focus on cyber security gaps and vulnerabilities within the security sublayer, a key management scheme that is non-broadcast but also feasible within an operational scenario, is determined. Finally, the effectiveness of the selected key management scheme on different threat models is assessed.

1.4 Chapter Objectives and Outline

To address the objectives stated above, we developed a theoretical network design framework for a multi-hop WSN that uses 6LoWPAN and IEEE 802.15.4 such that it can be deployed for tactical operations by the USMC.

This chapter discusses the following issues:

- Development of a command and control (administrative control) structure of the tactical WSN that incorporates node control, a unique defined/centralized routing model, and a selected keying mechanism for data confidentiality, authentication and integrity.
- Construction of a modified 6LoWPAN enabled IEEE 802.15.4 frame structure to incorporate the unique centralized routing model and selected keying mechanism.
- Enhancement of methods to aid in the deployment planning of the secured tactical WSN to prevent critical vulnerabilities.
- Simulation and evaluation of the proposed network framework against multiple attacks and testing for security robustness and energy conservation.

2 Related Works

There is an existing foundation of research in the literature that aims to achieve security and energy conservation within a WSN [4]. This chapter focuses on developing specific energy conserving security measures for a tactical WSN. To that end, we study specific tactical WSN architectures, routing mechanisms, the 6LoWPAN

frame structure and related cyber security mechanisms. In the following sections, we examine current research relating to these areas to provide a basis for the work presented in this chapter.

2.1 Architecture of a Tactical WSN

The architecture determines how the network is physically deployed and how the nodes are interconnected with one another. It also determines the types of devices to be used within the network and their functions. There has been work published on tactical WSNs that serve as a foundation for our work [13, 19]. A specific view of a tactical WSN architecture was taken in [13] for the deployment of the network in a large scale environment. The authors proposed an architectural design based on a cluster-tree network with multi-hop capability [13]. The cluster-tree design groups nodes into clusters, and a node is selected to act as a cluster head. The election of the cluster head is based on a nodes residual energy level [13]. The role of the cluster head is rotated among the nodes throughout the lifespan of the deployed tactical WSN, thereby conserving nodal energy. In addition, because the network is multi-hop, it allows the network to scale to a large environment.

The architecture model developed in [13] was also used in the model developed by [19], which applied the 6LoWPAN protocol. The protocol architecture for the deployed nodes was separated into five protocol layers. Each layer was examined, detailing the functions and techniques used within the layer. The IEEE 802.15.4 standard was determined to be a suitable protocol for the physical and MAC layers [19]. While detailing the adaptation and network layers, the authors of [19] assumed that the nodes would be randomly deployed, requiring the nodes to perform a self-organizing function through address auto-configuration. Other functions that were discussed include routing protocols, header compression, fragmentation, and energy saving [19]. Within the transport and application layers, the authors used a military application management entity that contains a military operations profile specifically defining parameters required for tactical deployment [19]. While [13, 19] provide architectural constraints for tactical WSN deployment, cyber security mechanisms and its relationship to energy consumption was not discussed.

2.2 Routing

Multiple methods of routing in a WSN exist, each of which is geared toward achieving a specific purpose such as energy conservation, low delay, or high throughput. A unique routing approach was taken in [4] where the objective was to conceal the sink node due to its high value as a target for an enemy attacker. In order to keep the sink node concealed, a routing algorithm was proposed that obfuscated the sink nodes location within the deployed network, reducing the risk of attack while preserving

the nodes energy levels. While the algorithm provided an anonymity mechanism to conceal the sink node within the network, a cyber security mechanism for the network communications using encryption and authentication was not provided. The proposed routing mechanism within [22] focused on the development of an energy efficient cross-layer load balancing and routing algorithm called EZone. With the application of the EZone routing protocol, the network lifetime was maximized; however, the EZone algorithm does not provide a mechanism for secure communications.

In [8], the Routing Protocol for Low Power and Lossy Networks (RPL), which uses routing control messages, was proposed as a routing mechanism for networks of low power devices. RPL is an end-to-end routing solution based on IPv6 communications and is specially adapted for the needs of specific types of traffic flow [20] and is capable of supporting control messages on multiple network architectures [8]. One of the disadvantages of RPL is that the protocol focuses on attacks that occur externally to the network rather than internally to the network [8]. The ability to provide a method of control over the routing scheme within the network is a form of security. This type of administrative control over the routing of data is a strategic decision that is applied within the work presented in this chapter.

2.3 6LoWPAN Frame Structure

In order to develop a feasible, secure design for tactical WSNs using 6LoWPAN, it is necessary to understand its frame structure. The composition of a 6LoWPAN frame was given in [9]. Given that the packet is reduced to a size of 127 bytes, some header information has been either removed or compressed. Two of the fields within the header that underwent significant compression were the IP addresses for the source and the destination nodes. 6LoWPAN offers two types of addressing modes where the IP address is either used in its entirety or is compressed. The compressed address mode offers the administrator an option to reduce the IPv6 address from 128 bits (16 bytes) to 16 bits (2 bytes), which saves 14 bytes per address, saving a total of 28 bytes [8]. The authors of [9] also demonstrate the implementation of a Compressed IPSecurity (IPSec) packet proposed in [16], which uses a UDP packet structure within the 6LoWPAN frame. The combination of the different compression methods and the implementation of compressed IPSec within the 6LoWPAN frame is the underpinning of our proposed frame structure in this chapter.

2.4 Security Mechanisms

A survey on security for the IoT was conducted in [8], which went into detail on the security vulnerabilities of the IEEE 802.15.4 standard in concert with the 6LoWPAN protocol. Within the IEEE 802.15.4 standard, the authors of [8] discussed different security mechanisms depending on a prescribed vulnerability. The model proposed

in [8] included the use of approved security modes, an access control list, and time synchronization to limit vulnerabilities, reducing the number of potential internal attacks; similar security mechanisms were used in [17]. The authors of [17] also provided greater detail on the potential attacks that could be used at each layer of the 6LoWPAN stack.

Encryption within the 6LoWPAN environment is a requirement in order to have an effective tactical WSN. Encryption has been addressed in the most recent release of the IEEE 802.15.4 standard [8, 17]. Multiple encryption modes are presented for use, and the challenge is for the researcher to determine what mode best fits the intended tactical WSN application. Along with encryption, there is a method to ensure data authentication called a Message Integrity Code (MIC). Knowing how encryption methods operate helps determine which one to use to defend against a variety of attacks. Advanced Encryption Standard-Counter with Cipher Block Chaining-Message Authentication Code (AES-CCM) is the suggested method within the 6LoWPAN standard [8]. Within the encryption method, an Initialization Vector (IV) is used. The combined fields within the IV provide a unique value to be used along with the encryption key, creating a unique encrypted payload for each transmitted packet. The keys for encryption are either public or private. Public and private shared keys both have positive and negative aspects. The application determines the type of key to be used.

An adaptation of the 6LoWPAN enabled IEEE 802.15.4 infrastructure developed in [16] incorporates the use of encryption over the typical IP infrastructure using IPSec. IPSec is the protocol suite used for IP communications to encrypt and authenticate IP packets. The IPSec suite is an immense protocol; thus, the full implementation of IPSec is inefficient within the 6LoWPAN enabled IEEE 802.15.4 infrastructure. The authors of [16] proposed a method to reduce the overhead within the IPSec protocol while maintaining the protocol's core capabilities. IPSec on the typical IP infrastructure uses two types of headers, the Authenticated Header (AH) and the Encapsulation Security Payload (ESP) header. The AH provides authentication of who sent the packet but the data is not encrypted, whereas in the ESP header, the data is encrypted, providing confidentiality to the transmitted data. Additionally, both the AH and ESP headers can be used together within the same packet. The information fields required for the AH and ESP headers are able to be incorporated within the already used header compression for the IP address and the Next Header field. The Next Header is an 8-bit field that identifies the next type of header immediately following the IPv6 header. The 6LoWPAN packet structure including the fields of the compressed header is shown in Fig. 2. The proposal by [16] further examines different types of encryption methods to be used within the ESP packet, including some of the security modes within the IEEE 802.15.4 standard.

Non-repudiation is another security measure that has been studied for 6LoWPAN networks. Non-repudiation ensures that the sender of a message cannot deny having sent the message. Non-repudiation is particularly useful in detecting and isolating compromised nodes [21]. A common technique to guard against non-repudiation is the use of a public key infrastructure (PKI) (also known as public key cryptography) which involves digital certificates to bind public keys with a user's identity. As it

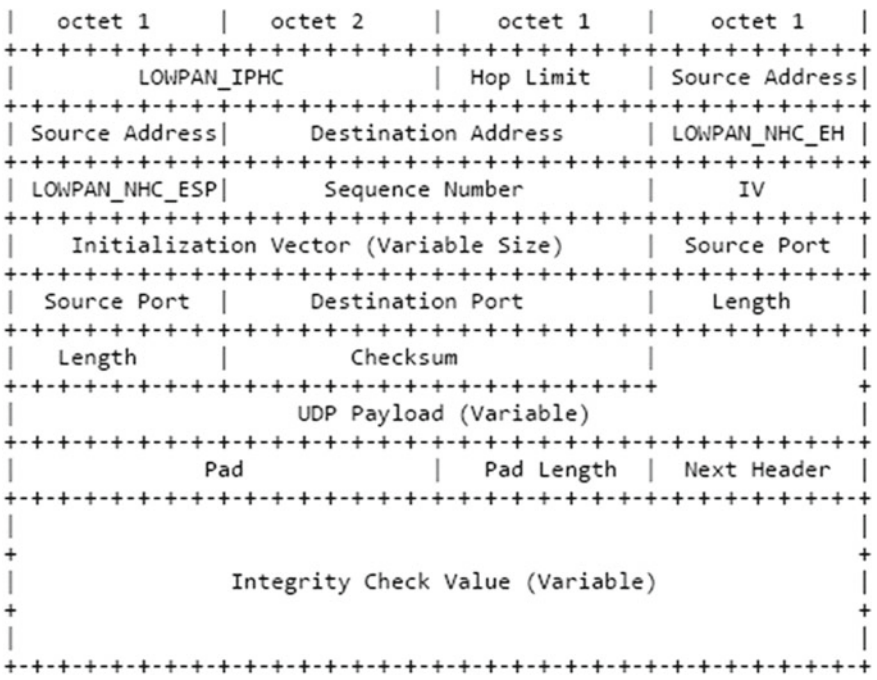


Fig. 2 Sample next header compression (NHC) compressed IP/UDP packet secured with ESP. *Source* [16]

is, PKI is considered to be not optimal for use in 6LoWPAN because it requires a significant amount of resource consumption and processing overhead [7]. The 6LoWPAN RFC does not discuss specific approaches to deal with non-repudiation. Despite the many advantages for PKI, it remains time and energy consuming on sensor nodes [11]. In this chapter, we focus on private keying rather than PKI. The keying mechanism will be further discussed in Sect. 3.1.9.

Location privacy of nodes in 6LoWPAN is also of interest. Anonymity schemes to hide the location and/or identity of the base station is very common [4, 5] in order to deal with adversaries performing traffic analysis. However, this topic is out of the scope of this chapter and was not considered the security architecture that is developed.

The final type of security mechanism to be discussed is Neighbor Discovery. Neighbor Discovery is a method of finding neighboring nodes through which the newly added node can route messages. Neighbor Discovery is a known vulnerability of 6LoWPAN networks [8]. The vulnerability lies in verifying if the neighbor is a node that is authorized to access the WSN. Multiple methods have been proposed to deal with this vulnerability, including a Lightweight Secure Neighbor Discovery for Low-power and Lossy Networks (LSeND) addressed in [16] as well as a variety of methods presented in [8]. Methods presented in [16] include RFC 6775-Neighbor

Discovery Optimization for 6LoWPAN, RFC 4861-Neighbor Discovery for IPv6, and an adaptation of RFC 3971-SEcure Neighbor Discovery (SEND). Given the complexity and known vulnerabilities of having a Neighbor Discovery protocol, limiting the capabilities of the network by disabling Neighbor Discovery mitigates both the complexity and vulnerabilities of the deployed WSN.

3 Theoretical Framework for Security Architecture

In this section, we discuss the theoretical framework for the design and implementation of a 6LoWPAN enabled tactical WSN using IEEE 802.15.4. This theoretical framework is developed with a focus on the cyber security mechanisms that are required for tactical deployment. We discuss (1) the network design, including the network devices used and their purpose, (2) the command and control (i.e., Network administration) parameters of the WSN, which detail specific tactical characteristics within the network (i.e., data routing and key management), and (3) the type of encryption and authentication algorithm used for the transmitted data. The network design, administrative privileges, and encryption and authentication of the network are the three main components of the security architecture for the tactical WSN. We then discuss the 6LoWPAN enabled IEEE 802.15.4 frame structure and the modifications that are made to the frame in order to deal with the cyber security considerations discussed. We also discuss recommendations for the deployment of the WSN for USMC tactical operations. Finally, the three types of cyber security attacks to be performed against the network are discussed. These attacks are used as an evaluation tool for the security architecture that is developed and is discussed further in Sects. 4 and 5.

3.1 Network Design

The proposed network design includes multiple elements, each serving a specific purpose. The elements included within the network architecture are as follows: master station (MS), base station/border router (BS), and sensor nodes. In the following sections, a description of each element is provided along with its intended use. In addition, the cyber security mechanisms associated with each network element are also provided along with assessments on attack mitigation. The proposed network architecture is shown in Fig. 3 and is based on a typical mesh network.

3.1.1 Master Station (MS)

The MS serves as the central node of the network, as depicted in Fig. 3. The proposed MS is a modified AN/MSC-77 (COC) currently used by the USMC but with

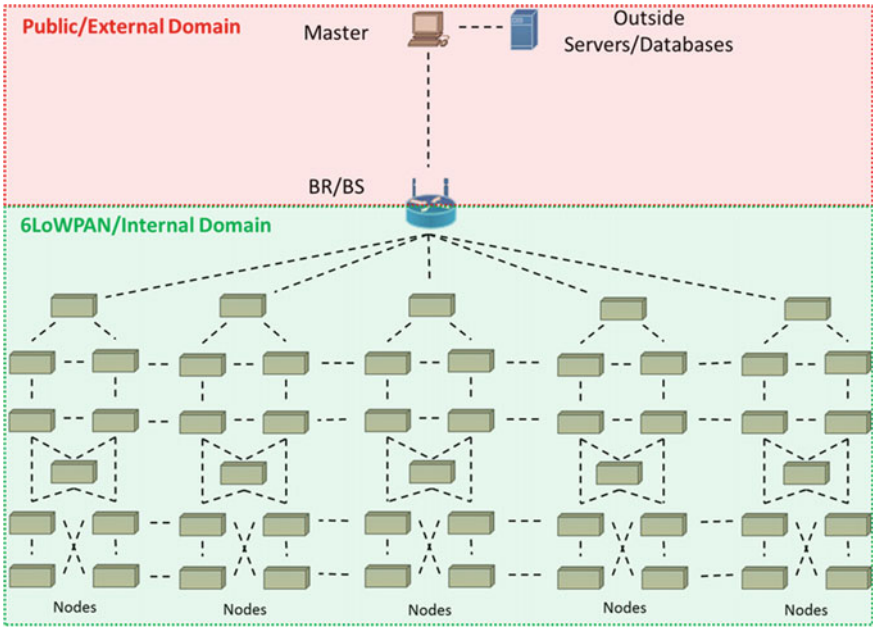


Fig. 3 Proposed 6LoWPAN network design

modifications. As discussed in Sect. 1.3, the typical USMC COC/MS unit must be within the line-of-sight of the WSN devices. Due to its large size, a typical COC/MS cannot be easily concealed. Rather than place the MS within the austere environment in which tactical WSNs are typically deployed, we position the MS in a structured, fortified military base, away from the tactical WSN region-of-interest, making it easier to protect. The proposed MS provides accessibility and administrative privileges to the sensor nodes while located at a safe, remote location away from the WSN. The MS has the ability to connect to each node within the internal domain since each element within the WSN supports two-way communication links. The MS can also connect to other servers outside of the WSN in order to disseminate data and populate other operator databases. When connected to other servers and databases, the data becomes accessible to remote operators, including those deployed within the WSNs environment. The MS also provides the operator a secure position to manage and control the WSN while extracting data from the sensor nodes. The flow of information discussed in this chapter does not require USMC personnel to rely on physically retrieving the stored data.

Cyber security mechanisms for the MS have already been developed and tested throughout the military (i.e., security mechanisms for the COC are well known). The MS will not be operating within the 6LoWPAN network environment but will be able to decipher the encrypted payloads sent to it from a 6LoWPAN device. As

a security measure, the encrypted payloads are the only method of communication between the nodes and the MS.

3.1.2 Base Station/Border Router (BS/BR)

The BS is the transitional element within the WSN that connects the 6LoWPAN/internal environment to the public/external environment. This is also commonly known as a sink node. The proposed BS is a secured router that transmits the data received from the sensor nodes into an external domain. The BS receives frames from the sensor nodes and removes unnecessary 6LoWPAN header information. It reassembles the payload into the compatible external network frame structure in order transit to the MS. The BS performs the same task in the reverse direction, removing unnecessary frame headers and adding the appropriate 6LoWPAN header to send the frame to the sensor nodes. Within the 6LoWPAN environment, the BS converts the addresses between the internal and external network environments since 6LoWPAN uses a modified addressing mode. The BS does not interfere with the payload because it is encrypted. The BS only contains the necessary encryption in order to connect to the MS; therefore, the frame payload to be transmitted remains secure. The actual transition of frames from one domain to the other is not the focus of this chapter and is not discussed at length.

The BS is restricted to 63 hops from the furthest node since the hop limit field within the frame structure consists of only six bits. The hop-limit field is further discussed in Sect. 3.3.4. Since the BS connects the WSN to an external network, it requires either a dedicated electrical supply, a generator, or robust battery supply as more power is needed to transmit a signal strong enough to reach the external domain. The BS is also able to withstand the harsh environments in which it is deployed and is much smaller than the COC since it does not need to have workstations available within the unit. The smaller size of the BS also allows it to be concealed more easily within the deployed environment. The BS should also contain anti-tamper technology to prevent physical modifications or reverse engineering of the device. Anti-tampering is already implemented on deployed USMC sensor devices [2]. We assume that the same tamper proof mechanisms can be implemented on the BS as well.

3.1.3 Sensor Node

The sensor nodes are the end elements. Each node is designed to attach to multiple types of sensors and to send data to the MS for compilation and analysis. The node is assumed to have the sensor capabilities as described in [2], including which sensors can be connected and which modes of operation are offered. The nodes have the ability to withstand the harsh environment and the capability to relay frames to the intended destinations. The node remains at its deployed location in order to maintain a static network. This facilitates our ability to sustain its continued connection to the

MS and to monitor each nodes energy use. The sensor nodes communicate with the MS only through encrypted payload routed through the BS. The nodes are deployed and contain anti-tamper physical security measures as noted in the USMC manual [2].

3.1.4 Attack Mitigation for Network Design

Vulnerabilities associated with the design of this tactical WSN include single points-of-failure and physical protection of the sensor nodes. The MS and BS are single points-of-failure to the WSN, and if removed, the network is no longer accessible and unable to be used. A denial-of-service (DOS) attack exploits this type of vulnerability. The MS has a greater impact on the WSN since it provides reachability, accessibility, and administrative privileges to the sensor nodes. The BS can be replaced by another BS without affecting the encryption or payload data transmission between the nodes and MS.

The vulnerability of the MS and BS is not the focus of this chapter, but the military does have similar devices in place today. Lastly, the physical protection of the nodes and BS remains an accepted risk. The deployed nodes will be able to detect an enemy approaching since the sensors can detect threatening events. The ability to sense potentially hostile events allows the sensor nodes to report suspicious data before becoming compromised, and if compromised, the node is removed by the MS to prevent further corruption within the WSN. In the end, the node is still able to perform the job it was deployed to do by detecting the enemy's presence even if it becomes compromised.

3.1.5 Command and Control (Administrative Control)

The administrative control aspects of the WSN are critically related to its functionality as well as the implementation of its cyber security mechanisms. These mechanisms are controlled by the MS. The control mechanisms of the MS include node control, centralized routing, and keying mechanisms. Using a centralized entity, such as the MS, to perform these functions, we limit attacks on the network.

3.1.6 Node Control

The MS maintains a directory of all of the networks connected to the BS as well as the sensor nodes within each network. The sensor nodes in the network are controlled by the MS via encrypted payload. The encrypted payload contains information that includes when the node provides real-time coverage or when the node stores detected events and transmits them in bulk at a later time [2]. Since all of the control messages are encrypted within the payload, the encryption provides the ability to securely control each node. The control of each node is limited to the MS. The MS serves as a centralized controller of all network functions. This centralization of control

removes the need to send an individual to make a modification to the BS or sensor for a new task; instead the node can be adjusted immediately from the MS. In the event a node is compromised, the MS can remove the node from the network as well as reset the node to delete its cryptographic and routing information, preventing the enemy from obtaining any data. The ability to remove the node either before, during, or after it has been compromised is a mitigating factor to the risks of a node deployed in a hostile environment.

3.1.7 Centralized Routing

As has been stated, the MS contains a directory of all nodes connected in the network. The MS implements centralized routing functions by controlling each sensor nodes routing table. In the network, data is either routed upstream, from the sensor node to the MS, or downstream, from the MS to the sensor nodes. For upstream data, each sensor node is only able to route data to two neighboring sensor nodes; each node has a primary path to the MS through one neighbor and a secondary path to the MS through the second neighbor. Exceptions to this include sensor nodes that are directly connected to the BS or when a sensor node is deployed to a location in which only one other node (neighbor) is within its wireless range. The sensor node also performs in the same manner downstream.

The centralized routing scheme allows the MS to add new nodes to the network by adding the new nodes address information into the neighboring nodes routing table via the encrypted payload. The MS also adds the necessary routing table information to the new node during the network setup. This method prevents the need for a neighbor discovery protocol, which is noted as a vulnerability within 6LoWPAN [8, 17]. The MS can also remove compromised or expired nodes by removing the node from the neighboring nodes routing tables; thus, any data sent from the obsolete node(s) is not routed.

An example of the centralized routing scheme of deployed sensors at an intersection is shown in Fig. 4. An intersection was used as an example since these devices track not only personnel but tanks or other manned vehicles [2]. The sensors are not limited to deployment at an intersection as they may also be deployed along a perimeter of a base or along a path of intended traffic. If an anomaly within the traffic flow occurs, it may mean an impending attack or an attack by the enemy is already underway within the area. To provide full coverage of an intersection, sensors are placed on each side of the road. The primary and secondary routing paths are marked, with every node having a secondary path except for the nodes with a direct link to the BS.

As discussed within the description of the MS, administrative control of the nodes is performed by the MS via encrypted payloads. If a node were to become compromised, the MS edits the nearby nodes routing tables via the encrypted payloads and then removes the compromised node from the network. An example of a compromised node and the adjustments made to the primary and secondary paths of the surrounding devices is shown in Fig. 5. In Fig. 5, node 17 is the compromised node.

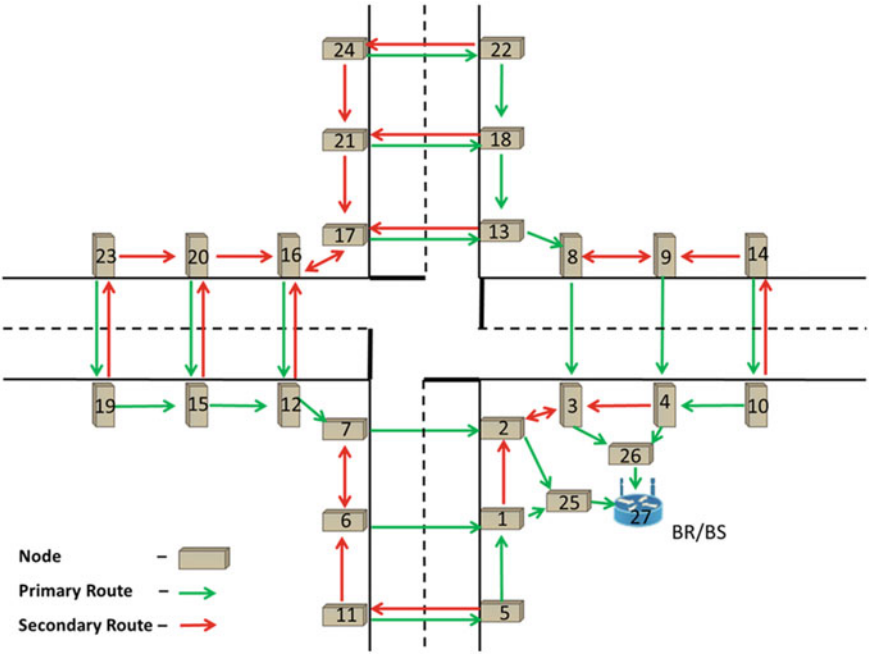


Fig. 4 Centralized routing scheme that depicts the primary and secondary paths for each node

The centralized routing scheme is adjusted to not allow a frame to be transmitted to or from the compromised node (node 17) as it is no longer in the routing tables of any node within the WSN. The routing adjustments depicted in Fig. 5 include nodes 12, 13, 16, 18 and 21. Each of these nodes loses its secondary routes, thereby limiting its ability to transmit to only one other node. Essentially, all paths that contain node 17 are removed, thereby isolating the compromised node. The adjustments in the routing path force some nodes to assume additional traffic loads, but the WSN is able to remain effective until the compromised node is repaired or replaced.

3.1.8 Data Transfer/Hidden Node Mitigations

A mechanism to transport data between nodes needs to be determined; however, the transporting mechanism to be used within an energy constrained environment must limit the creation of new network traffic. The authors of [16] use the User Datagram Protocol (UDP) as the basis of their packet structure, which in turn is the basis of the frame structure employed in this research. UDP also supports broadcast, multi-cast, and unidirectional communications. We focus on unidirectional UDP communications as it is better suited for the keying mechanism that is developed and discussed in this chapter. We discuss this further in the next subsections.

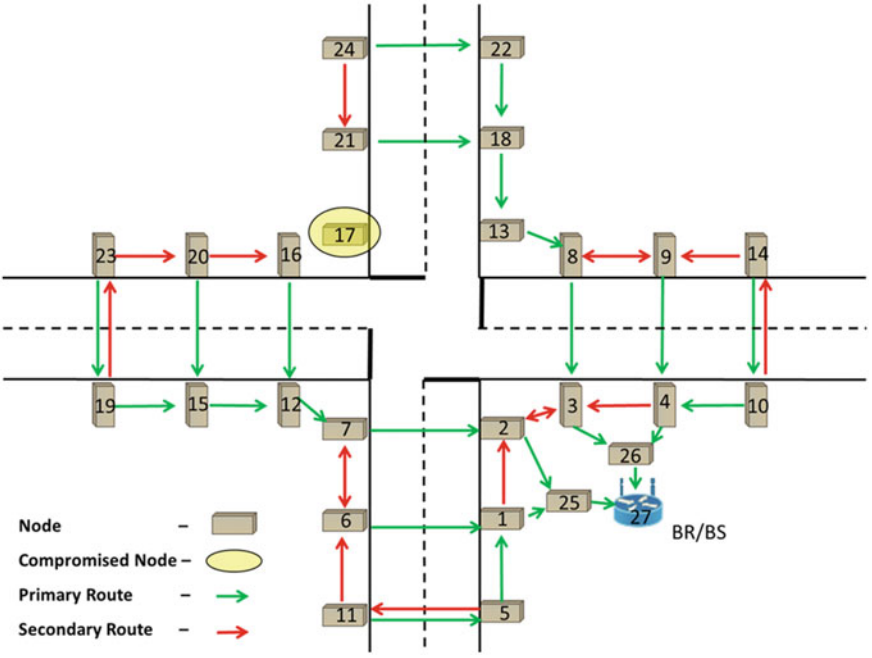


Fig. 5 Centralized routing scheme with a compromised node that depicts the changes to the primary and secondary paths of the surrounding devices

As mentioned above, UDP is a connectionless, best-effort mechanism used to route network traffic. This means that the sending nodes do not receive confirmation that the transmitted frame was correctly received [6]. To provide assurance that frames are received correctly within an unreliable transport protocol (i.e., UDP), we use a unique implicit acknowledgement detection mechanism. This unique implicit detection mechanism operates as described in the following paragraphs.

As previously mentioned, the communication links between the nodes are reciprocal; therefore, each node is able to see the next-hop transmission of the frame. This confirms that the frame was properly received without errors and forwarded. If the sending node does not see the next-hop nodes transmission, the sending node determines the frame needs to be retransmitted and performs the retransmission after the predetermined back-off period. If the transmitting node is forwarding to a relay node to funnel the traffic to the BS, a small random increment of time (seed number) is added to the back-off period. The predetermined back-off period is similar to carrier-sense multiple access-collision avoidance (CSMA-CA) with binary exponential back-off (BEB) in which the time between retransmissions doubles after each failed transmission attempt [14]. The seed number is applied in order to prevent repeated attempts at transmitting a frame at the same time as a neighboring node. If the sending node does not see the transmission of the frame from the follow-on node

after four re-transmissions, the sending node repeats the same process with the node designated in the routing table as the secondary route.

The primary advantage of using an implicit acknowledgment detection mechanism is the ability to ensure that a frame was received by the next-hop node without the extra network traffic required to set up sessions or send feedback control messages (acknowledgement frames) as used by other types of protocols. Essentially, this implicit acknowledgement detection mechanism takes advantage of the listening capability that each node has to detect transmissions within its transmission range. A disadvantage to this mechanism is the possibility of the originating node detecting the transmission of the follow-on node, which is transmitting a previously received frame from another node. The receiving node then uses the transmitted frame by the follow-on node as confirmation of a successful receipt.

This implicit acknowledgement detection mechanism cannot be applied to the nodes next to the BS as a result of the BSs transition and subsequent transmittal of the frame into the public domain. The neighboring nodes of the BS are not able to detect the follow-on transmission the BS makes since it is beyond the nodes capabilities. With the assumption that the BS has a dedicated power supply, the BS is continuously awake and ready to receive any frame sent to it. All of the traffic within the network is then funneled into the nodes surrounding the BS, which increases the network traffic. By limiting the BSs neighboring nodes to one transmission per frame, we reduce the possibility of congestion. With the increase in traffic density of the BSs surrounding nodes, hidden nodes also become a factor to consider.

In order to mitigate hidden nodes near the BS, extra relay nodes are used to funnel the traffic toward the BS, thereby removing possible hidden nodes from around the BS. A network implementation without the use of the extra relay nodes to funnel the network traffic is shown in Fig. 6. Nodes 1 through 4 can transmit to the BS; however, nodes 1 and 2 cannot see node 4 and nodes 3 and 4 cannot see node 1, creating a hidden node problem. If nodes 1 and 4 transmitted at the same time, a collision would occur, and the frame would be lost. The use of utilizing extra relay nodes within the vicinity of the BS to avoid hidden node problems is shown in Fig. 7.

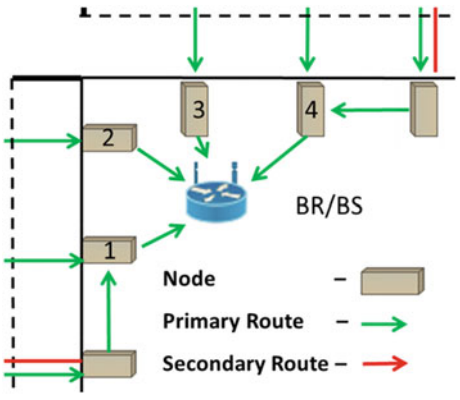


Fig. 6 Network design without the use of extra relay nodes

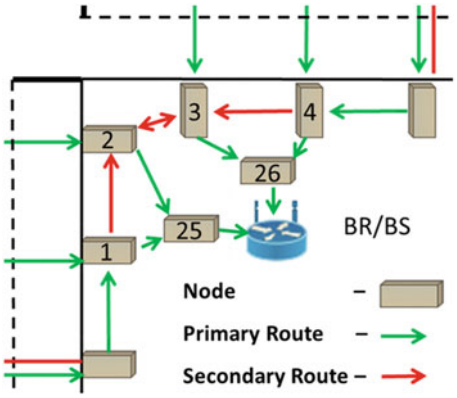


Fig. 7 Network design with the use of extra relay nodes (nodes 25 and 26) to illustrate the remedy for the hidden node problem

Within the network design and prior to deployment, the BS is moved further away from nodes 1 through 4 to allow for the addition of nodes 25 and 26 to the network. The BS is then far enough away to not be affected by nodes 1 through 4 and can only see nodes 25 and 26, while node 25 can only see 1, 2, 26 and the BS and, similarly, node 26 can only see nodes 3, 4, 25 and the BS. Nodes 25 and 26 are now functioning as a relay/sensing node and can see each other within the network. Since the extra relay nodes can see each other, the nodes do not transmit at the same time, preventing collisions from occurring at the BS. Also since nodes 25 and 26 are extra relay sensing nodes, they are able to concentrate on sensing the area around the BS to provide further physical security to the high value device.

One final modification was made to the delay in time between the first and second transmissions of a node. The first transmission is typically used to wake-up the receiving node, which affects the implementation of the BEB protocol. Traditionally, the BEB waits zero or one time slot before attempting a first transmission. Thereafter, the wait time doubles (two slots, four slots, eight slots, etc.) before each re-transmission. In our implementation, the BEB is modified as follows. A node waits 6.0 ms after the first transmission before trying to re-transmit. This time was adapted from [17]. The transmitting node does not know if the next-hop node is awake, thus this time permits the next-hop node to wake-up in preparation to receive the incoming frame for the second retransmission. The BEB begins after the second transmission but with an initial back-off time delay of 10.0 ms which then doubles after each successive unsuccessful attempt to transmit to the follow-on node. Thus, in our implementation, back-off times between the first and second transmissions do not follow the typical BEB delay of zero or one time slots; doubling of wait time does not occur until after the second transmission.

Each node has the ability to store frames that cannot be successfully forwarded. After a period of time, the node repeats the same process to forward the stored frame. By waiting a period of time without retransmitting, the node does not add to network congestion, thereby reducing possible collisions or delays. This is a capability already

demonstrated within [2] in its ability to store all events and transmit them at a specified time given by the network administrator.

3.1.9 Keying Mechanisms

A keying mechanism is needed to protect the information being transported between the nodes and the MS. Multiple keying mechanisms may be implemented, such as a private keying mechanism, in which each node has its own unique key, or a public keying mechanism, in which each node has both a private key specific to each node and a public key that is shared between all of the nodes. As was discussed in Sect. 2.4, PKC/PKI is a time and energy consuming protocol for WSNs. Previous research from [8] determined that after evaluating the energy requirements of PKI, private keying is the most efficient method to encrypt the payload. Public keying was also determined to have vulnerabilities, including an increase in transmissions throughout the network to update the keys. The focus of this chapter is to evaluate a non-broadcast key management method; therefore, individual frames are also used to disseminate each key update, creating more transmissions and, in turn, increasing energy use.

With the use of a private key, each node has a unique key that is only shared with the MS. As mentioned previously, the BS does not have any of the encryption keys shared between the nodes and the MS, but the BS does have a separate keying mechanism shared with the MS. The external network and, specifically, the keying mechanism for the external network are already in use in other areas within the military and is not the focus of this chapter.

3.1.10 Attack Mitigation for Administrative Control

Administrative control allows for mitigating factors if the WSN has nodes that are attacked via man-in-the-middle (MITM) or DOS. Each of these attacks requires an individual or remote device to be near the WSN, but the attacker is detected prior to performing the attack due to the sensing capabilities of the sensor node as previously mentioned above [2]. The centralized routing scheme prevents the intruder from further infecting and draining the rest of the networks power resources. Use of a keying mechanism also protects the data during transportation over the network; therefore, MITM or spoofing attacks are not able to change any of the data nor are they able to eavesdrop.

3.2 Encryption

Multiple encryption algorithms based on private keying are available. The algorithms include the Advanced Encryption Standard (AES), Elliptic Curve Cryptogra-

phy (ECC), and RSA. Each encryption method is authorized by the National Security Agency (NSA), which sets the encryption standards for the Department of Defense and establishes key lengths for various classification levels [3]. The most recent IEEE 802.15.4 standard lists eight security modes ranging from no encryption (one mode) to different versions of AES (seven modes) [8]. As a result, AES is used as the keying mechanism in the work presented in this chapter. The seven modes of AES boast different levels of encryption and authentication. Since the devices being used can be located within a hostile environment and are interacting with government networks, the highest levels of security are required within the WSN; therefore, the data must be encrypted and authenticated. To meet these requirements, AES-CCM with 128 bit keys is used as the keying mechanism to provide confidentiality, integrity, and authentication. It is shown that the selected encryption method also protects against MITM and spoofing. The keying mechanism is further discussed in the next section.

3.3 6LoWPAN Enabled IEEE 802.15.4 Frame Structure

The proposed 6LoWPAN enabled IEEE 802.15.4 frame structure is shown in Fig. 8 and is based on the structure defined in [16] with header compression schemes. In this work, we modified the frame structure to incorporate the previously discussed cyber security mechanisms. The fields within the frame structure are defined in the following subsections.

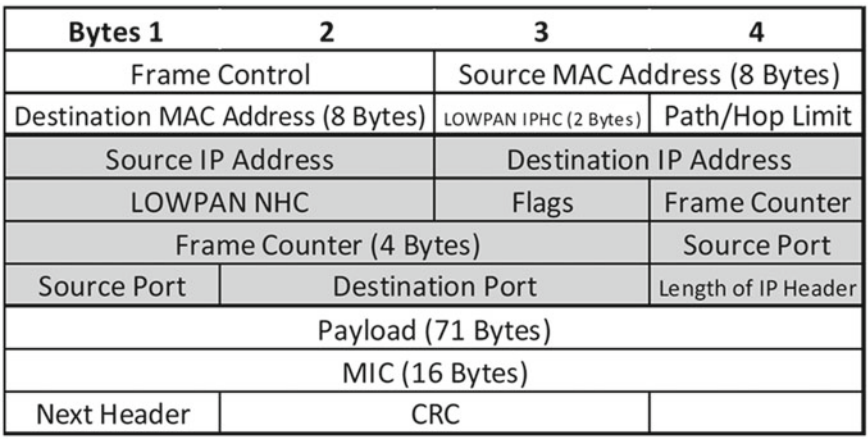


Fig. 8 Modified 6LoWPAN frame structure

3.3.1 Frame Control (2 Bytes)

This field has been defined by the 802.15.4 standard [9].

3.3.2 Source MAC Address/Destination MAC Address (8 Bytes Each)

This field has been defined by the 802.15.4 standard [9].

3.3.3 LoWPAN IPHC(2 Bytes)

This field has been defined by RFC 6282 [10] with no changes made.

3.3.4 Path (2 Bits)/Hop Limit (6 Bits)

The proposed centralized routing mechanism defined in this chapter limits the direction each frame can take to reach the BS from the transmitting node. This mechanism is a modification from the proposed frame structure in [16]. We use the Path/Hop Limit field in the following manner. The Path/Hop Limit field is a total of eight bits (one byte). Within the Path/Hop Limit byte, the first two bits are used to help the MS determine if there is an issue with a node routing frames. Specifically, the first two bits are used individually to determine whether the frame was transmitted over a primary or secondary route. The second bit is used only by the source node. In the event the primary route is used to send the frame, the bit is 0. If the secondary route is used, then the bit is 1. The same method is applied to the first bit and is used by all nodes except the originating node. When the MS receives the frame, it is known if a node was not able to transmit to a designated primary node. Depending on the modes of operation selected, the MS may be able to determine which node may be off line or compromised instead of waiting for a response or detection. The final six bits limit the number of hops a frame can take to $2^6 - 1$, or 63 hops. Limiting the number of hops to 63 does not present an issue since the nodes adjacent/neighboring to the BS are not able to support a large network of nodes. In other words, we want to maintain energy efficiency within the network. By reducing the possible number of hops within the network, we provide battery relief to those nodes, particularly those near the BS, that have to transmit data regularly.

3.3.5 Initialization Vector (16 Bytes)

The IV is used to help protect against replay attacks and is also used in the CCM process to encrypt the payload [8]. The IV is shown as the shaded portion of the frame in Fig. 8. The IV is a combination of unique identifiers, described as follows:

- Source IP address/destination IP address (2 bytes each): the addresses are in the compressed 16-bit mode, thus incurring smaller overhead as described within [16].
- LoWPAN NHC (2 bytes): this field is defined by RFC 6282 [10] with no changes made.
- Flags (1 byte): this field is reserved for the originating source and each bit is designated by the administrator of the WSN.
- Frame Counter (4 bytes): This field keeps track of the sequential order of frames sent; therefore, a separate sequence number field outside of the IV is not required.
- Source Port/Destination Port (2 bytes each): These fields have no changes or compression modifications [16].
- Length of IP Header (1 byte): This field is reduced to one byte since the maximum frame size is 127 bytes.

To prevent replay attacks, the frame counter (which is part of the IV) is used. A replay attack is when a valid message is maliciously or fraudulently repeated. The objective is to ensure sequential freshness of frames. The frame counter prevents a replayed message from being accepted by the receiver and ensures that the frame that has arrived is not a replayed one. The frame counter is incremented after each transmission and cannot wrap to 0. If a neighbor node receives a transmission with a frame counter that is less than or equal to the last received frame counter, the packet will be discarded as a replay transmission [refs: mac security and secutiy overhead analysis in ieee 802.15.4 wsn, thwarting attacks on zgibee-removal of the killerbee stinger].

3.3.6 Payload (71 Bytes)

The payload is the amount of data that can actually be transmitted. The data is encrypted, providing confidentiality during data transmission using the combination of the IV and the AES-CCM 128 bit key.

3.3.7 Message Integrity Code (16 Bytes)

To provide authentication and integrity, a MIC is created within the AES-CCM mode of encryption and is attached to the end of the frame. The MIC is a hash unique to the packet and is used to verify that no changes were made to the original message. The MIC provides another layer of protection against any attack that tries to inject or change data being transmitted.

3.3.8 Next Header (1 Byte)

This is used in higher layers and remains unchanged [16].

3.3.9 CRC (2 Bytes Each)

This field has no changes or compression modifications [16].

3.4 Deployment of Nodes

The deployment of the WSN is similar to [2] but with some modifications. It is proposed to first create the intended network, then connect the devices to the MS, and finally deploy the devices. Creating the network first requires setup of the key exchange for encryption purposes, the routing table to be loaded, and evaluation of ideal physical placements for the nodes. The administrator determines the physical location of the node within the designated network.

3.4.1 Key Exchange/Routing Table

After the network is designed and all of the routing tables have been constructed, the information for each node needs to be transferred to the node and BS. Each node and BS is physically connected to the MS for bootstrapping. The private key for the device and the constructed routing table is transferred to the node and BS. The routing table is transferred to the nodes by the MS to enable the nodes to connect to the network. The key exchange consists of a private key that is only shared between the MS and the node. The physical transfer of each unique key exists to prevent an enemy from gaining access to an entire networks information simply by obtaining the key from one node. By using a private key unique to each node, the enemy only has access to that nodes information. This also allows the MS to remove the node from the network by adjusting routing tables of surrounding nodes without compromising the rest of the network. Using a key also allows the MS to perform other security procedures, including resetting the node completely before it is compromised, thus preventing the enemy from obtaining information stored on the node.

3.4.2 Physical Placement

While connected, the MS is able to maintain a geographical map of deployed nodes and map the deployment of any new node. This helps determine if the pending placement of the new node is able to connect to surrounding nodes. This is critical to the deployment of the WSN since it helps track enemy movements and position. Since the MS is also able to compile data from all deployed nodes, this allows the data to be utilized by more entities. For example, warning messages can be automatically disseminated to surrounding units if a certain threshold is met, alerting them of enemy activity within the area.

3.5 Proposed Attacks

Three different types of attacks are conducted in the simulated environment: spoofing, MITM, and DOS. Each attack uses a different method of execution and exploits different vulnerabilities within a network, but as previously reviewed, the implemented cyber security mechanism can either prevent or lead to the detection of an attack, which triggers the deployment of mitigation methods that maintain the integrity of the network. The attacker in each of the following scenarios is assumed to be an experienced hacker. Under this assumption, the attacker knows what MAC and IP addresses to use within the injection frame as well as the key used for the cyclic redundancy check (CRC); however, it is assumed that knowledge of the pre-shared key between the node and the MS is not known.

3.5.1 Spoofing

Spoofing is simulated by a rogue node pretending to be a legitimate node within the network. To conduct the attack, a device is required to be within range of the nexthop node. The frame is then transmitted to the next-hop node to increase the probability that it will traverse the network. Operating under the assumption that the attacker is experienced, the frame successfully reaches the MS. The MS then examines the frame and determines the frame is not authentic since the MIC does not match, therefore dropping the frame. The MS also records the node the frame was from for future comparison. After multiple frames have been received, the MS can analyze the results and determine if there is a rogue node within the network. The actions of the rogue node can be mitigated by removing the node that was initially spoofed, denying all traffic sent by either of the nodes.

3.5.2 MITM

A MITM attack is simulated by placing a node between two network nodes. When a network node transmits a frame, the attacking node intercepts and changes the data within the frame before transmitting the original frame to the next-hop node. The frame continues to transit the network until it reaches the MS. The MS examines the frame and determines it is not authentic while recording the result within its logs. Since a MITM attack affects more than one node, it is assumed that the MS can rapidly determine that a rogue node has invaded the network. The neighboring nodes affected by the attacking node can then be removed from the network.

3.5.3 DOS

A DOS attack is the disruption of services to a specific node within the network. To simulate a DOS attack on a node, a continuously transmitting rogue node is placed near a network node. This keeps the affected node constantly receiving from the rogue node, which prevents the network node from transmitting any detected events or forwarding any frames from other nodes. Using the path indicator bits within the frame, the MS can determine where the disruption is occurring and modify the networks routing tables to mitigate the disabled node.

4 Experimental Setup

In this section, we discuss the experimental setup used to perform network simulations using MATLAB. These simulations test the efficacy of the cyber security mechanisms implemented in the theoretical framework. We start by describing the characteristics of the sensor nodes used in the simulations, the phases of operation of each sensor node and the frame parameters. We then describe the network parameters in relation to the simulation. Lastly, we discuss the simulation program including the different WSN modules that were developed, the required user files, and the simulation logs.

4.1 Sensor Parameters

The sensor-node attachment for the purpose of detecting events is not the focus of this chapter; however, some detection parameters are assumed in order to have a fully functioning network simulation. The sensors in all simulations are the Magnetic Intrusion Detector (MAGID) described within [2]. This sensor is designed to detect large vehicles such as tanks and small vehicles as well as individuals; although, detection ranges vary depending on the power level being used and the element size.

During the simulations, sensors are set on a low power setting and placed along a two-lane intersection. The deployment of the nodes with an attached MAGID are shown in Fig. 9. The low power setting has a maximum range of 15.0 m for the detection of vehicles, whereas the maximum range for the detection of individuals is 4.0 m [2]. The ranges are illustrated by the ellipses in Fig. 9 to exhibit the detection areas of the deployed sensors. The extra relay nodes near the BS are not shown in Fig. 9 since the focus of the figure is to illustrate sensor coverage.

As mentioned in Sect. 3.1.8, the extra relay nodes are used to funnel the network traffic to the BS and prevent hidden nodes. In addition, the extra relay nodes provide coverage to the WSNs only single point of failure, the BS. A close up view of the BS, demonstrating how the extra relay nodes function together as a sensing node and the resulting coverage is shown in Fig. 10.

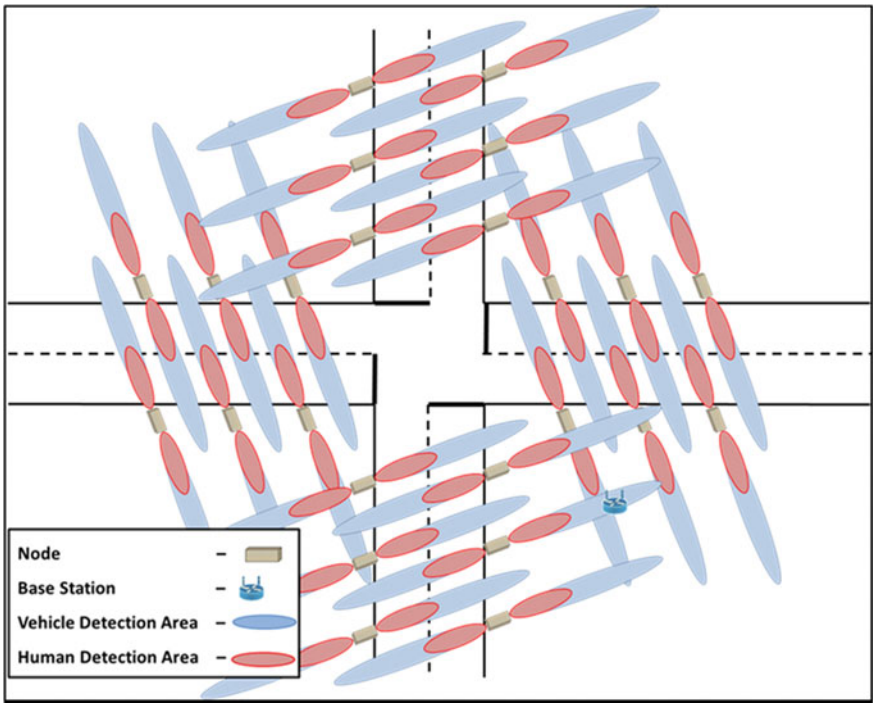


Fig. 9 MAGID deployment coverage of an intersection

4.2 Node Characteristics

Each node transitions between multiple phases of operation/execution. The various phases of node operation along with the amount of power (in mW) that is used at each phase and the duration of each phase (in ms) are outlined in Table 1. These metrics identify the amount of time and energy used by the node to perform its functions in the simulations. The duration times shown in Table 1 were adopted from [18]; however, we rounded these times to the nearest millisecond because the simulation program we developed is based on one millisecond increments.

The Receive-Transmit (RX-TX) requires the node to switch from receiving to transmitting in order to transmit the frame and then switch back to receiving; thus, the execution of transmitting a frame requires two RX-TX phases, one to switch from receiving to transmitting and two to switch from transmitting to receiving. The RX-TX phase within [18] has a period of 0.4 ms, which totals 0.8 ms during the transmission and receiving process of a frame. Since our simulation programs smallest increment in time is 1.0 ms, 0.8 ms is rounded to 1.0 ms and is used to represent the two RX-TX switching phases detailed in Table 1.

To complete a frame transmission, the node must execute the following phases, totaling a period of 7.0 ms: CSMA, RX-TX switch, transmit, and RX-TX switch.

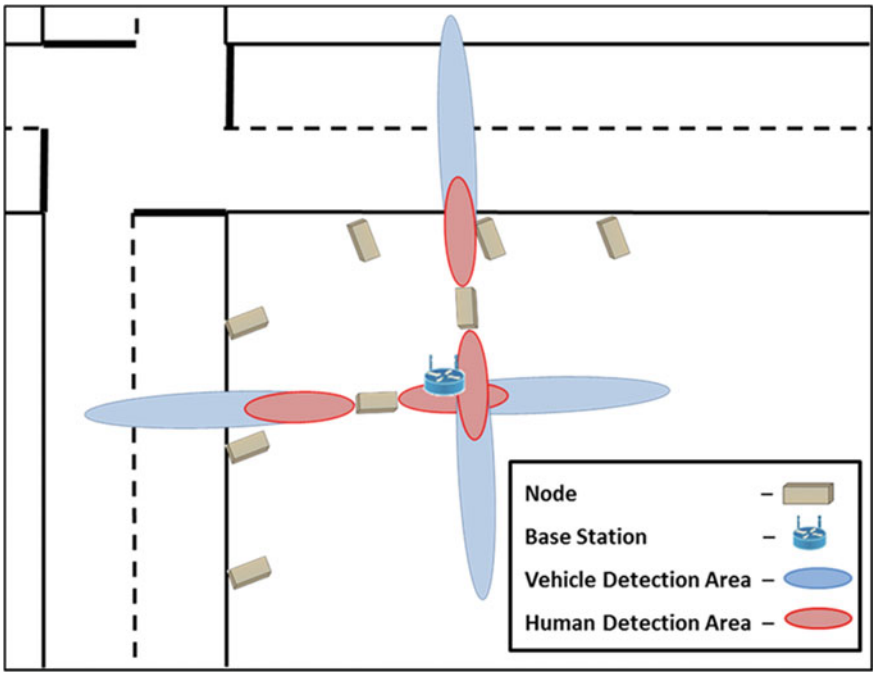


Fig. 10 Close-up view of the MAGID deployment coverage of the extra relay nodes used to funnel traffic to the BS and avoid hidden nodes

Table 1 Phases of node operation. Adapted from [18]

Phase	Power draw (mW)	Duration (ms)
Wakeup	20	1
Pre-process	24	4
CSMA-CA	72	2
Transmit	90	4
Receive	72	4
RX-TX switch	54	1
Post-process	24	1
Waiting	72	Varies
Go to sleep	20	1

Multiple logs record the status of the node throughout the lifespan of the network. One log only records each time a node transitions from one phase to another, while another log tracks the status in which the node is currently, and a third log is used to record the nodes time in phase by the millisecond. The nodes status logs are critical since they are used by multiple modules within the simulation program. The nodes status logs include the expiration time for the phase in which the node is to prevent the

node from performing/entering another phase. In most simulation programs, detailed status change logs and current node status logs are not available to the researcher, limiting a researcher's ability to troubleshoot any errors. As this network is meant to be deployed by the USMC, the ability to troubleshoot is imperative; thus, we provide the researcher with the pertinent logs. These logs are further discussed in Section G of this chapter.

Within [18], encryption was not addressed. We assumed encryption was not incorporated into the power and time metrics given in Table 1; therefore, we incorporated the time and power draw for encryption into the simulation. In [12], metrics for various brands of sensor nodes performing multiple methods of encryption with varying key sizes are given. To give the best results, we chose to use the TelosB platform since it is more power efficient [12]. Within the TelosB platform, we used the power draw and times obtained from the implementation of AES 128 to match our WSN model. The time and power draw metrics used in the simulation with AES 128 were 3.77 ms and 7.47 μW, respectively.

The nodes within the simulation are only awake for the time period in which there is activity. If a node does not have any activity for 25.0 ms, it transitions to the sleep phase. The duration of 25.0 ms was chosen to compensate for transmissions occurring when the node is processing a received frame from a node that is hidden from the transmitting node. The duration of 25.0 ms accounts for the time between the third and fourth transmissions, CSMA-CA, and both the RX-TX switch phases between actual transmissions of a frame, allowing the follow-on node to see the retransmission before entering its sleep phase.

There may be instances when the transmitting node cycles through both the primary and secondary routes without a confirmation of the next hop node transmitting the frame. In this situation, the transmitting node stores the packet and waits 1.0 s plus a seed number between 1 and 200. This provides enough time for the network to clear any data congestion, transmit all the frames to the BS, and allow all the nodes to enter their sleep phase.

4.3 Frame Parameters

As with a node, the frame also transitions through multiple phases. The phases include creation (unprocessed), transmission, processing, and completion. A log is also maintained, which tracks the status of the frame as it transitions between phases. Within the log, a phase expiration is attached to each nodes entry and coincides with the duration metrics used for the nodes.

4.4 Network Parameters

As previously discussed, the network implements a modified CSMA-CA BEB protocol with an additional seed number added to the BEB for frames transiting from the neighboring nodes to the BS. The primary focus of this section is to discuss the modified CSMA-CA BEB and the parameters used. We also discuss the addition of the seed number for the nodes forwarding traffic to the BS via an extra relay node.

4.4.1 Modified CSMA-CA BEB

CSMA-CA BEB is a protocol that determines a set time period between transmissions of the same frame to avoid collisions with other transmissions from other nodes. Once the initial time period is determined, each retransmission doubles the previous time period between the transmitted frames [14]; however, this model is used within the IEEE 802.11 standard and assumes that each node is awake and receiving the frame [14]. Within the simulated network, when the nodes are no longer forwarding or sensing other events, they go to sleep to conserve energy. This sleep feature requires the transmitting node to send a frame to wake-up the next hop node. Even though it only takes 1.0 ms for a node to wake up, the transmitting node must take into account that the next-hop node might already be awake and receive the frame successfully; thus, the time the transmitting node must wait before beginning the second transmission process is 6.0 ms, which permits receipt of the next-hop nodes transmission.

After the second transmission, the typical BEB takes place with the initial backoff period of 10.0 ms, then 20.0 ms after the third transmission and 40.0 ms after the fourth transmission. If forwarding of the frame has not been detected after completing the fourth transmission, the transmitting node switches the frames next-hop destination to the secondary route. The transmitting node then performs the same sequence again while simultaneously adjusting the designated path bits within the header.

4.4.2 Seed Number

The CSMA-CA BEB protocol also provides a method to alleviate network congestion in the node by retransmitting at different set time intervals [14]. To prevent the occurrence of two nodes transmitting at the same time near the BS, a seed number between 1.0 and 4.0 ms is added to the back-off time. The seed number creates disparity between the back-off times between transmitting nodes. With the application of the seed number, neighboring nodes that initially transmit at the same time has a smaller probability of retransmitting at the same time since each node applies a random seed number to the BEB. This helps mitigate congestion caused by collisions as the network begins to taper to the BS.

5 Simulation Results and Analysis

An assessment of the proposed 6LoWPAN WSN is needed in order to validate the effectiveness of the implemented cyber security mechanisms. While the focus is on the cyber security mechanisms within a 6LoWPAN enabled WSN, it must be anticipated that these WSNs can be deployed to different types of environments. The different types of environments create varying levels of network traffic. We simulate these different environments using four scenarios that mimic the varying levels of traffic density. The scenarios developed within this section mimic vehicles transiting an intersection at different speeds. These scenarios then allow the researcher to determine the effects of the implementation of the proposed cyber security mechanisms within a variety of simulated network environments with each simulation listed as a trial.

The network topology used in the simulation is shown in Fig. 4. A reference number is given to each node in the figure. The reference number provides a simple method for the researcher to reference a node within the program and within the user created files. There are four scenarios simulated, each representing a specific speed of vehicle traversing an intersection. The first scenario represents vehicles traveling at 25 miles per hour (mph), the second at 35 mph, the third at 45 mph, and the last scenario represents vehicles traveling at 65 mph. We chose these numbers to simply show a range of speeds, from slow (25 mph) to fast (65 mph). Each of the four scenarios are initially simulated with no attacks occurring. These trials act as the normal conditions of the tactical WSN, creating a baseline to compare the results from simulations that incorporate attacks performed on the WSN.

The nodes selected for each attack remain the same for all scenarios to allow for comparisons between the different network environments. The total number of trials conducted for each scenario is as follows: five trials with no attack, five trials with a spoofing attack on node 24, five trials with a spoofing attack on node 16, a DOS attack on node 5, a DOS attack on node 25, and an MITM attack between nodes 7 and 2. In total, each scenario has six different network implementations with five trials per implementation, resulting in a total of 30 trials for each scenario. This results in a total of 120 trials for the program. Due to space, we show and discuss only a subset of the results obtained.

In the following subsections, each attack is discussed and analyzed. An analysis of each of the attacks is displayed from the view of the MS with the implemented cyber security mechanisms in place; these mechanisms lead to either the detection or mitigation of the attack. The power draw for each node, which is not analyzed by the MS, is also discussed. Within the analysis slight variations within the results may be observed due to the implementation of frame errors. The frame errors are caused by frames being dropped or not properly received and can occur undetected on the last hop to the BS.

5.1 Spoofing Results

The purpose of the spoofing attack is to test the efficacy of the MIC security mechanism added to the IEEE 802.15.4 6LoWPAN enabled frame. The two nodes imitated in the spoofing attack are nodes 16 and 24. Node 16 was chosen since the surrounding nodes have a higher traffic density than a node on the edge of the network. Node 24 was selected since it is on the edge of the network with limited network traffic flow and is one of the least protected nodes in the WSN. In order to detect a spoofed node within the WSN, the MIC security mechanism is used to authenticate the frames received by the MS.

The number of spoofed frames detected by the MS and sent by the rogue node imitating node 16 is shown for Scenarios 2 and 3 in Figs. 11 and 12, respectively. As expected, the number of detected, spoofed frames varies between trials due to the activity of the receiving node. The number of detected, spoofed frames also varies between scenarios. This was determined through further analysis of the logs maintained during each trial. It is observed that fewer frames were injected as the speed of the vehicles being simulated increased.

We calculate and analyze the rate of the frames injected. The average number of frames injected for the five trials performed for Scenario 2 was 43 (Fig. 11), while 38 frames were injected in Scenario 3 (Fig. 12). The rate for frame injection per second was computed using

$$R_{FI} = \frac{FI_{avg}}{T_{Last\ Frame} - T_{First\ Frame}} \tag{1}$$

Since the simulation program performing the analysis of the MS can only tell if the frames being processed are spoofed, the logs were examined to determine

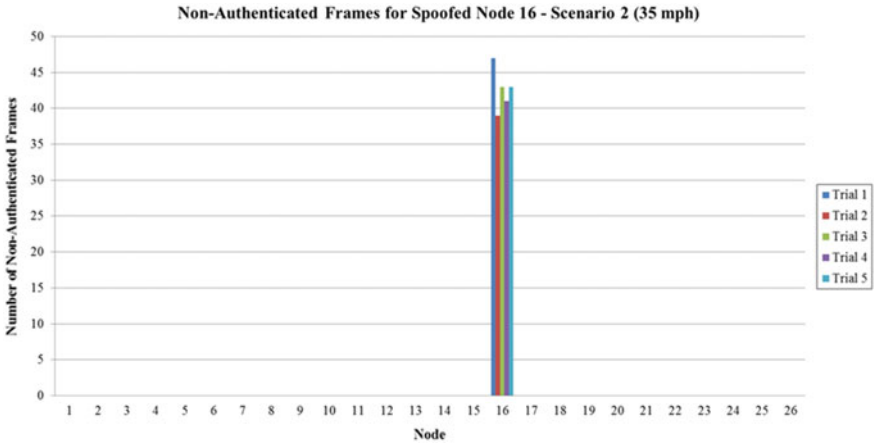


Fig. 11 Number of non-authenticated frames received by the MS in each of the five trials for scenario 2 simulating a spoofing attack on node 16

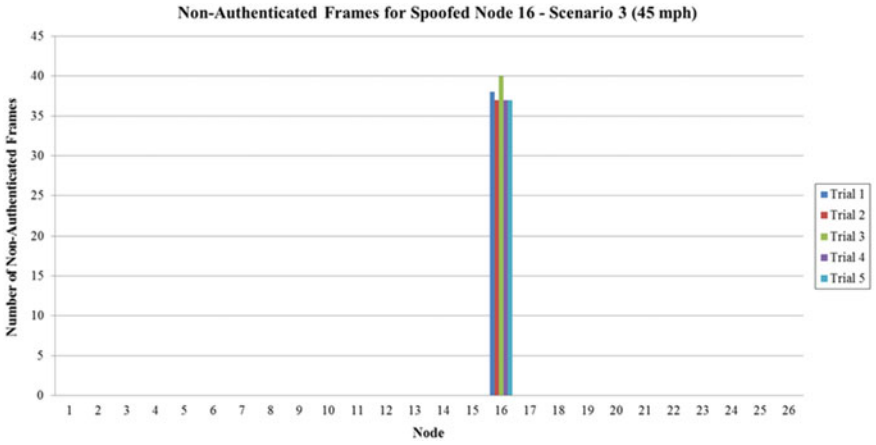


Fig. 12 Number of non-authenticated frames received by the MS in each of the five trials for scenario 3 simulating a spoofing attack on node 16

if a spoofed frame was not detected and processed. After reviewing the logs and determining which frames were spoofed, 100% of the spoofed frames received by the BS within the program were detected and denoted as non-authenticated by the MS. Within the logs, it was also noted that not all spoofed frames were received, as a small percentage were either lost or received in error by the BS.

The number of frames detected by the MS that were sent by the rogue node imitating node 24 is shown for Scenarios 2 and 3 in Figs. 13 and 14, respectively. Nodes 16 and 24 share similar characteristics in the slight variation of injected frames

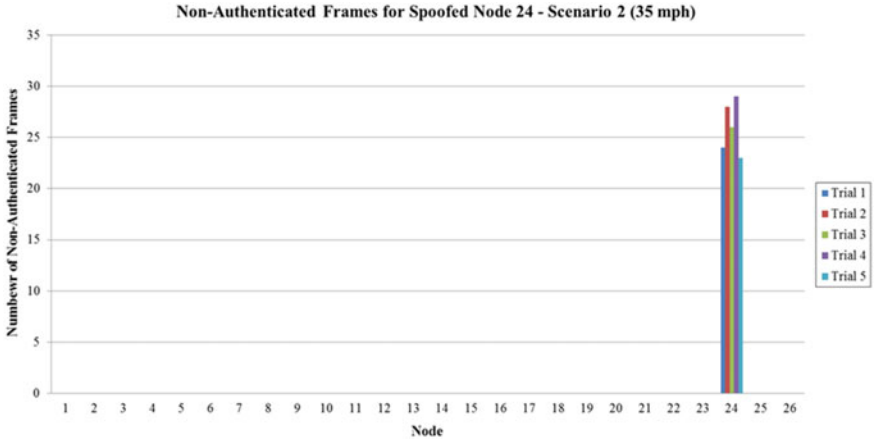


Fig. 13 Number of non-authenticated frames received by the MS in each of the five trials for scenario 2 simulating a spoofing attack on node 24

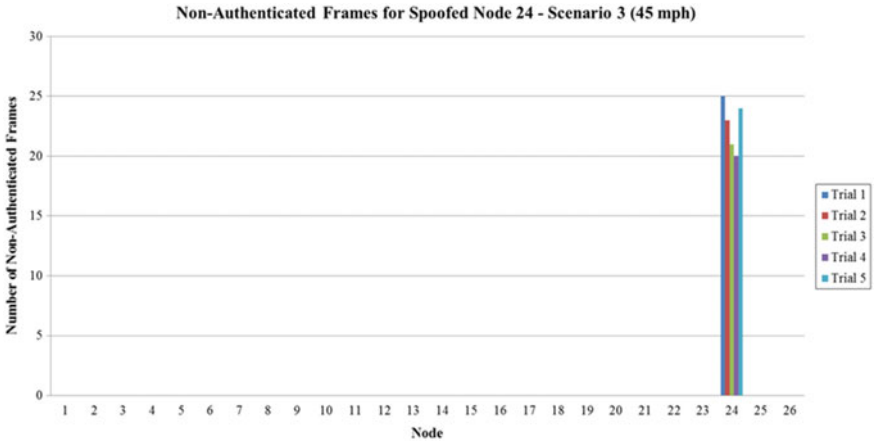


Fig. 14 Number of non-authenticated frames received by the MS in each of the five trials for scenario 3 simulating a spoofing attack on node 24

between trials for a given scenario as well as between the scenarios; however, since node 24 is on the edge of the network and has a low density of network traffic, the node is asleep more than node 16. This explains the difference in the average number of frame injections. The average number of frames injected for the five trials performed on Scenario 2 was 26 (Fig. 13) while Scenario 3 s average was 23 (Fig. 14). The calculated rates were also similar to those for node 16; for node 24 for Scenario 2, R_{FI} was 0.221 frames per second, resulting in an injected frame every 4.52 s. The R_{FI} for Scenario 3 was 0.238 frames per second, resulting in an injected frame every 4.20 s.

The ability of the MS to perform an analysis on received frames to determine if there is a possible spoofing attack within the WSN, as well as to determine which node to remove from the WSN to prevent further attacks, is illustrated in Figs. 11 through 14. The analysis focused on the use of the implementation of the MIC security mechanism to authenticate valid frames sent by the nodes. Variations were also visible within the results between the trials and scenarios being run, but they were expected and were accounted for in the network logs. Overall, the MIC security mechanism is able to provide data integrity by allowing the MS to authenticate each received frame. In addition, while the increase in the number of frames transmitted has an impact on the power draw, it is minimal, keeping the WSN functional.

5.2 DOS Results

The DOS attack was used to determine if the centralized routing scheme and the use of the path indication bits could detect an attack or incapacitated node. The two nodes

for the DOS attack are nodes 5 and 25 due to their differing characteristics within the WSN. Node 5 is on the edge of the network with limited network traffic flow and is also able to be physically accessed undetected by an individual due to the MAGID coverage. Node 25 is one of the extra relay nodes deployed near the BS and handles half of the traffic within the network. An attack on Node 25 demonstrates the ability of the WSN to remain reliable if an extra relay node is incapacitated. Specifically, a DOS attack on node 25 validates the networks ability to utilize the secondary route of the centralized routing mechanism. In addition, it also validates the WSNs capability to accommodate an increased traffic load. Scenario 1 and Scenario 4 are used to compare the DOS results.

5.2.1 Original Hop

Our analysis of the DOS attack examines the centralized routing mechanism executed by the MS. Specifically, we first look at the number of frames the originating node failed to successfully send along the primary route. These results are shown in Fig. 15 for Scenario 1 and Fig. 16 for Scenario 4. Nodes 1, 2 and 11 are observed to have a significant increase in number of frames transiting to the next-hop node from the originating node. The increase for nodes 1 and 2 occurred during the trials in which there was a DOS attack on node 25, and the increase for node 11 occurred during the trials in which there was a DOS attack on node 5. This is expected since the primary route for node 11 is through node 5, and the primary route for nodes 1 and 2 is through node 25.

A closer examination of Fig. 15 indicates that a limited number of frames from nodes 7, 8, 16, and 17 also took secondary next-hop routes. These anomalies can

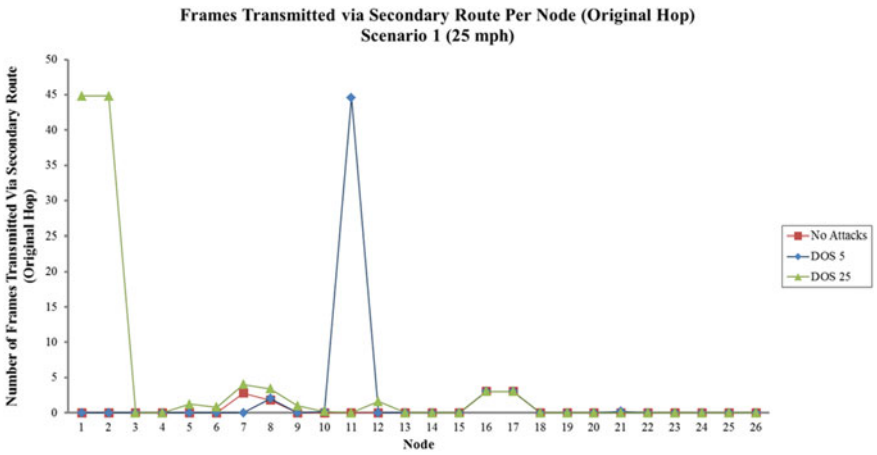


Fig. 15 Frames transmitted via secondary route per node (original hop) in scenario 1 for no attacks and DOS attacks at nodes 5 and 25

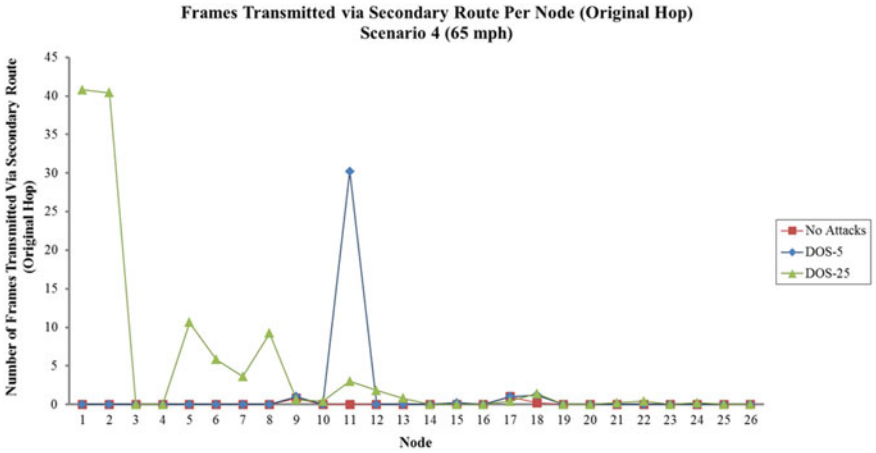


Fig. 16 Frames transmitted via secondary route per node (original hop) in scenario 4 for no attacks and DOS attacks at nodes 5 and 25

be attributed to a small amount of congestion within the network as it occurs even when no attacks are executed as denoted by the results in red in Fig. 15. Also shown in Fig. 16 are the same anomalies that occurred in Fig. 15. As shown in Fig. 16, a moderate number of frames from nodes 5, 6, 7, 8 and 11 also took secondary next-hop routes. This is attributed to the increased congestion from not only the DOS attack on node 25 but also the increased density of the traffic due to the faster detection rate. The network was not able to clear the congestion quickly enough, and the congestion began to spread to nodes two to three hops from the attacked node. Even though there was a greater amount of congestion in the network, it remained secure and continued to reliably deliver frames from affected nodes with near real-time precision.

The analysis presented in Figs. 15 and 16 is based on the implementation of the centralized routing mechanism with the utilization of the path indication bits. The path indication bits alerts the MS that the frame was not able to successfully send the packet along the primary route from the originating node. The use of the secondary route is an indication of possible congestion or a node malfunctioning, causing the WSN to operate in a non-optimal manner. Furthermore, this increases the power draw of multiple nodes and possibly causes more network congestion. Using the information gained from the path indication bits, we see that the MS adjusts the centralized routing mechanism for optimization.

5.2.2 Follow-On Hop

The DOS attack is further examined through the analysis of the number of frames that were unsuccessfully transmitted along the primary route by the follow-on nodes. These results are shown in Fig. 17 for Scenario 1 and Fig. 18 for Scenario 4. As shown

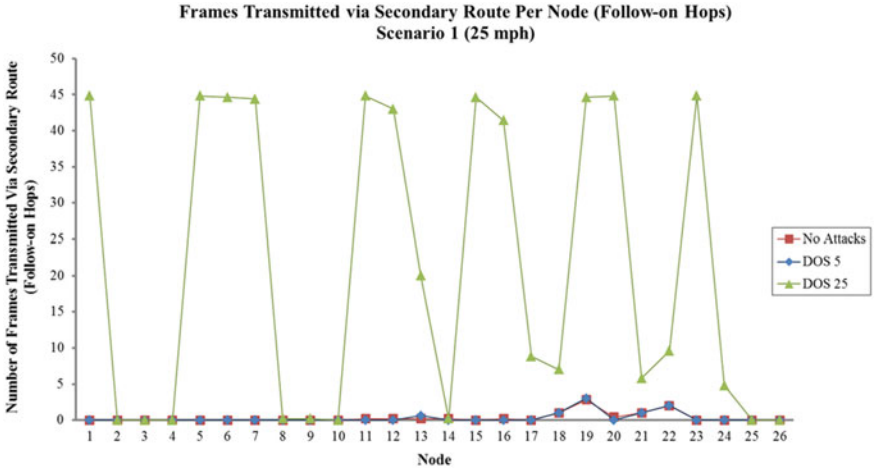


Fig. 17 Frames transmitted via secondary route per node (follow-on hops) in scenario 1 for no attacks and DOS attacks at nodes 5 and 25

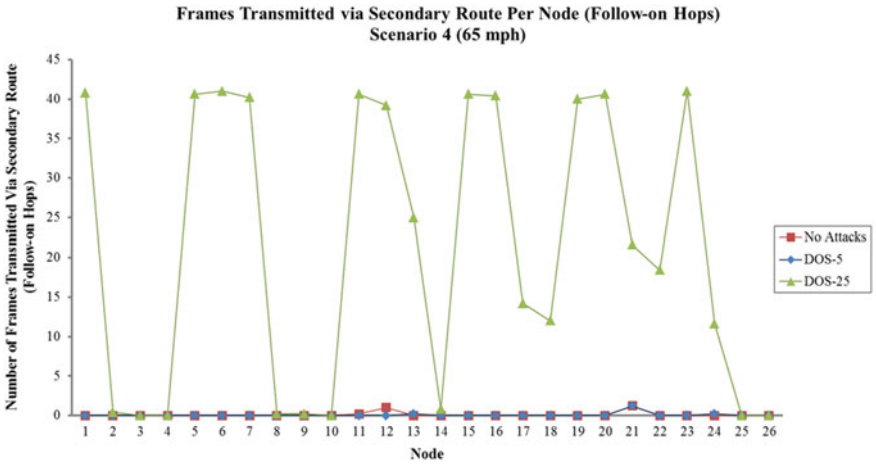


Fig. 18 Frames transmitted via secondary route per node (follow-on hops) in scenario 4 for no attacks and DOS attacks at nodes 5 and 25

in Fig. 17, a DOS attack on node 5 does not produce any significant changes in terms of the number of unsuccessful transmitted frames. This is expected since node 5 is only a primary route for node 11 and is not a primary route for any other nodes within the network; however, the analysis of the DOS attack on node 25 shows a significant increase in nodes that sent frames that utilized a secondary route at a follow-on node. The significant increase in the utilization of secondary routes at follow-on nodes is expected since node 25 is the primary route for half of the WSN to the BS. Minor escalations at nodes 13, 17, 18, 21, 22, and 24 can be explained

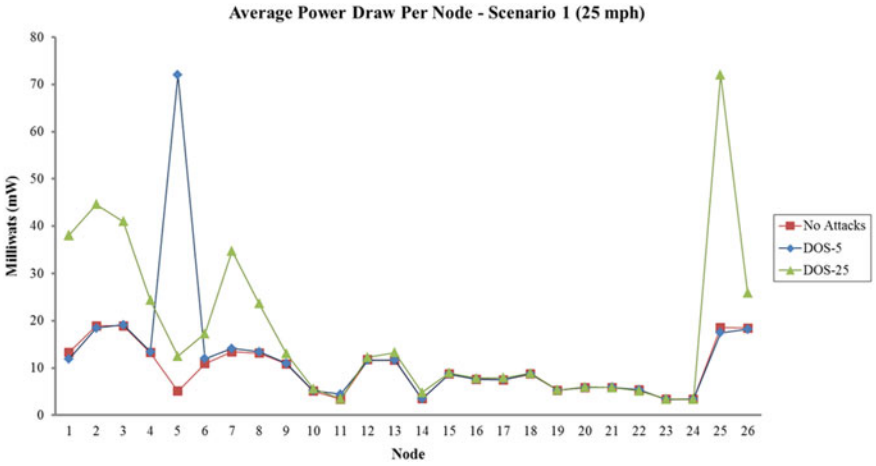


Fig. 19 Power draw per node in Scenario 1 for no attacks and DOS attacks at nodes 5 and 25

by congestion experienced from the surge of network traffic utilizing the secondary routes. This is more evident in Fig. 18 due to the increased density of network traffic in Scenario 4.

The analysis presented in Figs. 17 and 18, like Figs. 15 and 16, are possible due to the implementation of the centralized routing mechanism with the utilization of the path indication bits. The secondary route information data provides an additional awareness of the WSNs status, which the MS uses in order to determine if and/or how much congestion is within the WSN. The combination of the follow-on hop data as well as the original hop data provides a near real-time status of the WSN to the MS.

5.2.3 Power Draw

The average power draw for each node during the simulations of Scenarios 1 and 4, in conjunction with either a DOS attack on node 5 or a DOS attack on node 25 as well as no attacks, are shown in Figs. 19 and 20. As shown in Fig. 19, there are minimal changes in the power draw between no attack and the DOS attack on node 5 trials except for node 5. The increase in power draw for node 5 is expected because node 5 is constantly receiving a signal from the rogue node performing the DOS attack. It is also observed in Fig. 19 that the DOS attack on node 25 affects all of the nodes up to two hops from the BS as well as nodes 7 and 8, which are three hops from the BS. This is attributed to the increase in network traffic causing congestion, which in turn increases the power draw of the affected nodes. Similar characteristics are also seen in Fig. 20, with the same nodes having an increase in their power draw for both attack simulations.

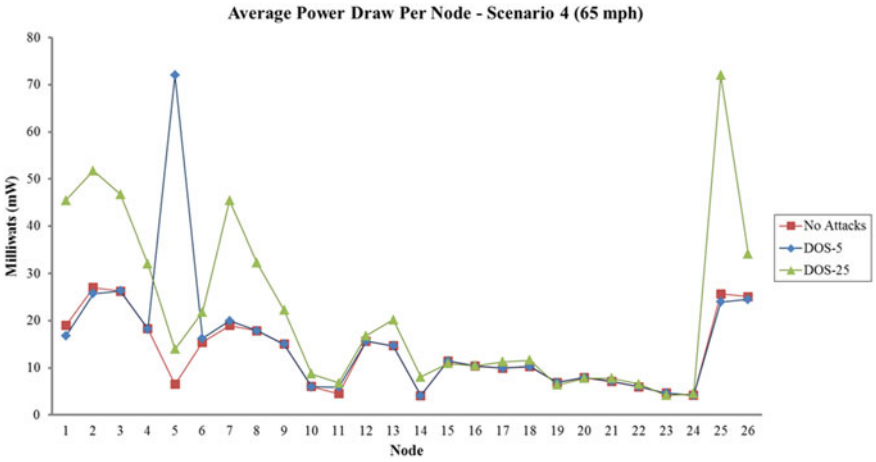


Fig. 20 Power draw per node in Scenario 4 for no attacks and DOS attacks at nodes 5 and 25

The results shown in Figs. 19 and 20 have a strong correlation to the results shown in Figs. 15 and 16. The nodes from Figs. 15 and 16, which utilized secondary routes to transmit the frame from the original hop, correspond to the same nodes with an increase in power draw in Figs. 19 and 20. The increase in the power draw for the surrounding nodes is expected since it requires the originating node to first transmit to the primary route four times prior to transmitting to the secondary route. After further examination of the results in Figs. 19 and 20, we find that the nodes in Fig. 20 had a much larger power draw than those in Fig. 19. As mentioned before, Scenario 4, shown in Fig. 20, simulates an environment in which detections occur at a rate approximately two times greater than Scenario 1, shown in Fig. 19. These results are expected since the nodes have less time to sleep and spend more time awake, which increases the power draw for the nodes.

5.3 MITM Results

Similar to the spoofing attack, the analysis performed on the MITM attack is focused on the implementation of the MIC security mechanism but also the centralized routing mechanism using the path indication bits. The wireless connection between nodes 7 and 2 was selected as the point of attack since it has a high density of traffic affecting a large portion of the network but not the entire network. The power draw of the nodes was not impacted as much as the Spoofing and DOS attacks. Shown in Fig. 21 is the analysis of the power draw for no attacks and an MITM attack between nodes 7 and 2. The differences between the MITM simulations and no attack simulations are negligible since the power draw for each node is the same.

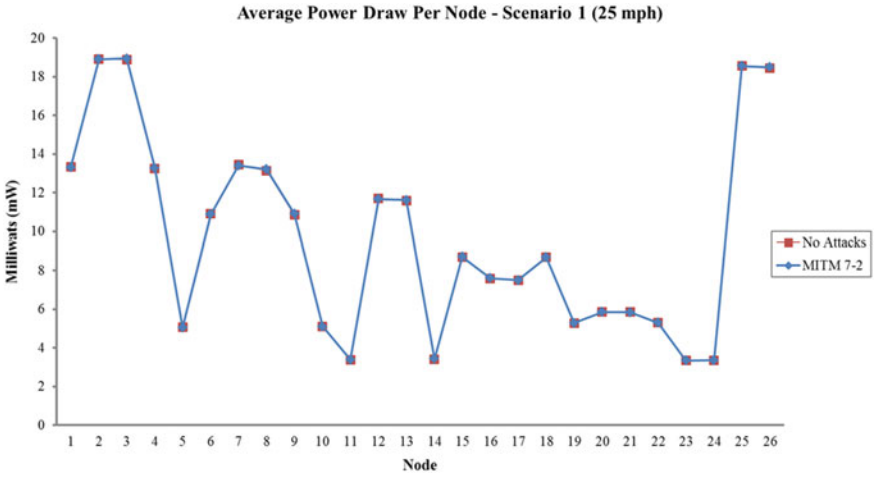


Fig. 21 Average power draw per node in mW when no attack occurs and during the MITM attack simulations between nodes 7 and 2

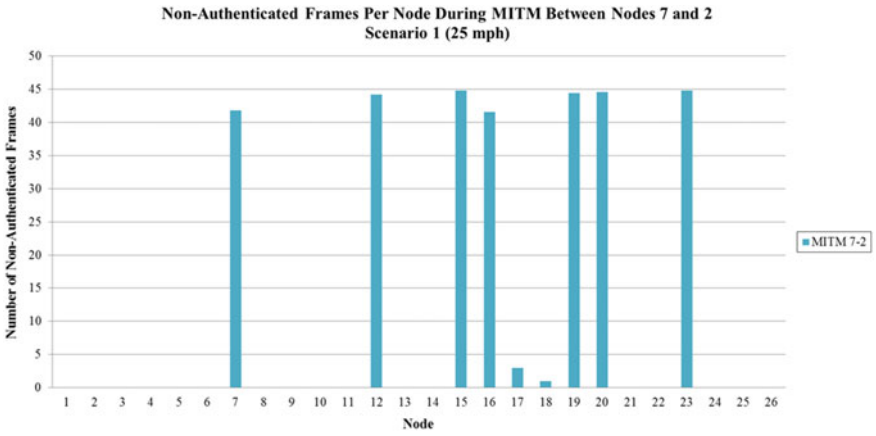


Fig. 22 Power draw per node in Scenario 4 for no attacks and DOS attacks at nodes 5 and 25

The average number of non-authenticated frames over the five conducted trials for Scenario 1 is shown in Fig. 22. Nodes 7, 12, 15, 16, 19, 20 and 23 have primary routes through node 7 to node 2, and these nodes had the most non-authenticated frames. The number of detections per node is 45 with a frame generated for each detection. A slight variation between the frames sent by nodes and the frames received by the MS is shown in Fig. 22. The variation is accounted for by either a node along a frames path utilizing a secondary route, resulting in the frame not transitioning through the affected nodes, or the frame being lost or dropped at the BS. The non-authenticated

frames from nodes 17 and 18 are due to congestion within the network and the utilization of the secondary route, which ultimately went through the affected nodes 7 and 2.

6 Conclusion

In this chapter, we discussed the implementation of the 6LoWPAN protocol for tactical WSNs and examined the need for 6LoWPAN in tactical WSNs used by the USMC in operational scenarios. Through the use of 6LoWPAN, our aim was to reduce the manpower required to maintain the tactical WSN by allowing the WSN to be managed from a remote, secure location. Ultimately, 6LoWPAN provides automation to the data flow by eliminating the need of an individual to physically retrieve data from the COC. The 6LoWPAN protocol, with the addition of necessary cyber security mechanisms, can be implemented and used by the USMC to boost the abilities of its current WSNs.

This chapter presented a comprehensive tactical WSN framework using 6LoWPAN that includes a hierarchical network design using defined network devices. The use of a structured/centralized network design allows for secure network reachability and accessibility. We implemented multiple cyber security mechanisms within the 6LoWPAN protocol. These security features included a centralized routing scheme, encryption, authentication and integrity. These features were applied/implemented into the 6LoWPAN frame structure. The combination of these various cyber security mechanisms and frame structure modifications create an effective and efficient tactical WSN that can be utilized by the USMC.

We evaluated our framework using MATLAB and tested it against three well-known attacks. Results obtained from the simulations focused on the effectiveness of the cyber security implementations. The use of the MIC provided integrity to the WSN by preventing the authentication of 100% of the frames received by the MS in either the spoofing or MITM attacks. The use of the centralized routing scheme ensured the WSN remained functional and reliable even when one of the two nodes connecting the BS to the rest of the WSN was disabled during the DOS attack. The implementation of indication bits within the modified 6LoWPAN frame structure enabled the MS to determine that there was congestion within the network resulting from either traffic density or an incapacitated node.

This chapter provided an effective and efficient tactical WSN using 6LoWPAN that can remain reliable and secure while deployed within harsh environments. We conclude that the developed cyber security mechanisms and network structure provide a foundation on which future tactical WSNs used within the USMC can be based. Our future work includes designing security algorithms for specific vulnerabilities of the 6LoWPAN network, including neighbor discovery and routing.

References

1. Sensor Mobile Monitor System An/MSC-77 Technical Manual, TM 09856A-10/1A
2. Unattended Ground Sensor Set AN/GSQ-257 Technical Manual, TM 09632A-OI
3. Agency, N.S.: NSA suite B cryptography. https://www.nsa.gov/ia/programs/suiteb-cryptography
4. Callanan, A., Thulasiraman, P.: Achieving sink node anonymity under energy constraints in tactical wireless sensor networks. In: Proceedings of IEEE CogSIMA, pp. 186–192 (2015)
5. Deng, J., Han, R., Mishra, S.: Decorrelating wireless sensor network traffic to inhibit traffic analysis attacks. Pervasive Mobile Comput. J. (Elsevier) **2**(2), 159–186 (2006)
6. Dulaney, E., Easttom, C.: Mastering TCP/IP-CompTIA Security+ Study Guide, 6 edn. Wiley
7. Goswami, S., Misra, S., Taneja, C., Mukherjee, A.: Securing intra-communication in 6LoWPAN: a PKI integrated scheme. In: Publication is IEEE International Conference on Advanced Networks and Telecommunication Systems (ANTS), pp. 1–5 (2014)
8. Granjal, J., Monteiro, E., Silva, J.S.: Security for the internet of things: a survey of existing protocols and open research issues. IEEE Commun. Surv. Tutor. **17**(Third Quarter), 1294–1312 (2015)
9. Hennebert, C., Santos, J.: Security protocols and privacy issues into 6LoWPAN stack: a synthesis. IEEE Internet Things J. **1**(5), 384–398 (2014)
10. Hui, J., Thubert, P.: Compression format for IPv6 datagrams over IEEE 802.15.4-based networks. Technical Report 6282 (2011)
11. Krentz, K., Rafiee, H., Meinel, C.: 6LoWPAN security: adding compromise resilience to the 802.15.4 security sublayer. In: Proceedings of ACM ASPI, pp. 1–10 (2013)
12. Lee, J., Kapitanova, K., Son, S.: The price of security in wireless sensor networks. Comput. Netw. (Elsevier) **54**(17), 2967–2978 (2010)
13. Lee, S., Lee, S., Song, H., Lee, H.: Wireless sensor network design for tactical military applications: remote large scale environments. In: Proceedings of IEEE MilCom, pp. 911–917 (2009)
14. Lin, Y., Hwang, R., Baker, F.: Computer Networks: An Open Source Approach. McGraw Hill (2012)
15. Olsson, J.: 6LoWPAN demystified (2014). http://www.ti.com/lit/wp/swry013/swry0.13.pdf
16. Raza, S., Duquennoy, S., Selander, G.: Compression of IPsec AH and ESP headers for constrainteed environments (2013). https://tools.ietf.org/pdf/draft-raza-6lowpan-ipsec-00.pdf
17. Rghioui, A., Bouhorma, M., Benslimane, A.: Analytical study of security aspects in 6LoWPAN networks. In: 5th International Conference on Information and Communication Technology for the Muslim World, pp. 1–5 (2013)
18. Siekkinen, M., Hiienkari, M., Nurminen, J.K., Nieminen, J.: How low energy is bluetooth low energy? Comparative measures with zigbee/802.15.4. In: Proceedings of IEEE WCNC Workshop on Internet of Things Enabling Technologies, pp. 232–237 (2012)
19. Song, H., Lee, S., Lee, S., Lee., H.: 6LoWPAN based tactical wireless sensor network architecture for remote large-scale random deployment scenarios. In: Proceedings of IEEE MilCom, pp. 1–7 (2009)
20. Thulasiraman, P.: RPL routing for multigateway AMI networks under interference constraints. In: Proceedings of IEEE ICC, pp. 4477–4482 (2013)
21. Vidyarthi, D.: Technologies and Protocols for the Future of Internet Design: Reinventing the Web. IGI Global (2012)
22. White, K., Thulasiraman, P.: Energy efficient cross layer load balancing in tactical multigateway wireless sensor networks. In: Proceedings of IEEE CogSIMA, pp. 193–199 (2015)

Data-Driven Detection of Sensor-Hijacking Attacks on Electrocardiogram Sensors

Hang Cai and Krishna K. Venkatasubramanian

Abstract In this chapter, we build a detector for identifying sensor-hijacking attacks that alter sensor readings in a **Wearable Medical Internet of Things (WMIoT)**. A sensor-hijacking attack targets medical devices in a WMIoT and makes them generate arbitrary user health state information. A sensor-hijacking attack is very dangerous because it can be mounted stealthily on unsuspecting WMIoT users. We focus on sensor-hijacking attack on one of the most commonly collected vital signs from a person, the electrocardiogram (ECG) sensor. Using sensor-hijacking attack to alter ECG measurements can have profound consequences for the user, as an adversary can easily make a person who is experiencing cardiac arrhythmia appear to be normal and thus cause immediate or long-term harm to their health. To detect sensor-hijacking-based alterations of the ECG measurements, our approach leverages the idea that multiple physiological signals based on the same underlying physiological process (e.g., cardiac process) are inherently related to each other, i.e., have common features. Any surreptitious alteration of one of the signals will not be reflected in the other *reference* signal (s) in the group. We describe an ECG alteration detector that uses arterial blood pressure (ABP) measurements as reference. The advantages of using a distinct reference signals are: (1) It does not require redundant ECG sensors to operate, (2) it adapts to the changing physiology of the user and does not depend on historical trends, and (3) it does not require instrumentation on other hardware modifications of the devices to work. In this work, we describe a temporal ECG alteration detector. Analysis of our detector shows promising results with over ∼97% accuracy in detecting even subtle ECG alterations for both healthy users and those with cardiac conditions, within 5 s of the occurrence of the sensor-hijacking attack.

H. Cai · K. K. Venkatasubramanian (✉)
Department of Computer Science,
Worcester Polytechnic Institute, Worcester, MA 01609, USA
e-mail: kven@wpi.edu

H. Cai
e-mail: hcai@wpi.edu

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_20

1 Introduction

Our society has been facing considerable challenges in recent years. Increasing traffic congestion, energy scarcity, climate change, and many other issues have taken a turn for the worse and need urgent attention. One such area is that of providing quality health care to people in a safe and effective manner. The healthcare system in most countries has increasingly come under pressure as the average age of their population increases and the number of elderly people swells. This will most likely lead to dire shortages of healthcare personnel, and, if left unattended, could result in a drop in the quality of medical care and a substantial increase in healthcare costs [39]. Technology can play a major role in alleviating this problem through the development of smart health monitoring systems.

In the last two decades, crucial technological breakthroughs have made it possible to miniaturize sensing, communication, and processing platforms that can be embedded as a part of larger systems/processes for providing real-time monitoring and feedback control services. Such platforms, deeply embedded in physical processes, are fueling the *Internet of Things (IoT)* revolution that we are all experiencing. The idea behind IoT is to incorporate intelligence in everyday objects/services in order to improve the efficiency of performing routine but crucial tasks. Examples include simple systems such as smart coffee pots that can detect the decrease in the temperature of the coffee and alert the user so that the coffee does not have to be unnecessarily re-heated; to complex ones such as closed-loop control of an individual's type 1 diabetes using artificial pancreas. *IoT systems in the form of wearable medical technologies, referred by us from now on as* **Wearable Medical Internet of Things (WMIoT) systems**, *can play a huge role in alleviating the problems of providing improved health care through the realization of continuous health monitoring*.

WMIoT systems are a collection of wearable medical devices to enable pervasive, long-term, real-time health management. A typical WMIoT has *sensing medical devices (aka sensors)* that collect, store, and communicate physiological (e.g., vital signs), activity (e.g., gait, sleep), and environmental (e.g., temperature, pollen-count, humidity) data from within and around the user it is deployed on, and forwards it to a *base station* entity, such as a smartphone or smart-watch for processing, visualization, and storage. WMIoT systems may also contain *actuating medical devices* that can provide therapies to the user based on the data collected by the aforementioned sensors. In general, the output of the medical devices in WMIoT can be used to understand the user's health state and provide appropriate treatments. The use of WMIoT for health monitoring provides several benefits:

- They provide the ability to monitor users in their "habitat," which allow caregivers to understand the causes of user's symptoms and thus provide personalized care.
- They allow for continuous monitoring of users thus reducing the overcrowding at healthcare facilities, while at the same time maintaining the standard of care.
- They allow very few caregivers to monitor a large body of users and therefore alleviate some of the effects of shortage of caregivers from aging populations.

- They allow for quick detection of medical emergencies and alerting of EMT, other caregivers, and the user's relatives.
- They can be extended, and indeed are already being extended, to provide closed-loop management of chronic conditions, thus minimizing frequent caregiver intervention.

Recent years have seen a dramatic increase in the number of wearable monitoring systems, particularly fitness trackers. Examples include Fitbit [10], Jawbone [12], and AppleWatch [4]. We view these devices as the first generation of WMIoT systems that are not medical devices but rather provide movement and in some cases heart-rate monitoring of the user based on in-build accelerometer and optical monitors, respectively. More complex systems are already available such as Empatica's E4 devices [9] and Sotera Wireless' Visi [46] that allow the user to monitor not only movement and heart rate but also electrocardiogram (ECG), continuous blood pressure, skin conductance, and body temperature. We expect this trend to continue in the future as we move from non-medical physiological monitoring to medical grade ambulatory physiological monitoring using wearable systems. Already several next-generation WMIoT platforms are available such as [3, 5, 6, 8, 13, 14, 16]. While the continuous sensing capabilities of these systems offer unprecedented opportunities for personalized care [44], the consequences of an attack on the same sensing capabilities of the WMIoT system can be devastating.

1.1 Sensor-Hijacking Attacks on Wearable Medical IoT Systems

Security is essential for WMIoT environments. Lack of security in WMIoTs not only harms user privacy, but may also harm the user's safety. Data security is also a legal requirement that WMIoTs have to fulfill due to Health Insurance Portability and Accountability Act (HIPAA) [11]. Though devices in the WMIoT can be attacked in many ways, in this chapter we focus on adversaries who mount what we call *sensor-hijacking attacks*. We *define* sensor-hijacking attacks as attacks that prevent sensing medical devices from correctly collecting and reporting the user's health state (e.g., reporting old or wrong physiological measurements). Sensor hijacking in WMIoT systems presents itself due to vulnerabilities in four basic areas of the system: (1) the communication channel, (2) the software and firmware update process, (3) the unprotected sensory-channel, and (4) from direct physical compromise.

The communication channel used by wearable systems is usually wireless in nature, for example, using Bluetooth Low Energy (BLE), Wireless Medical Telemetry Service (WMTS), MedRadio, or ZigBee. An insufficiently secured communication link within the WMIoT can allow adversaries, from *relative proximity* of the user (victim), to introduce bogus data, modify/suppress legitimate health data into the system. Such user health data manipulation can lead to erroneous evaluation of a user's health, untimely administration of treatment, or denial of service (DoS).

Communication links in wearable medical systems are rarely secured properly and consequently, recent years have seen a plethora of attacks newly identified for wearable medical devices that exploit the wireless communication channel. Examples include the reverse engineering and replay attacks on pacemakers [30] and insulin pumps [23, 35] that allow the adversary to not only know the current state of the user's health but also modify it.[1]

In addition to communication attacks on wearable medical devices and their data, sensor-hijacking vulnerabilities also exist in the software/firmware update process. Medical devices in WMIoT systems are complex entities that need to be updated from time to time in the field to provide new features, fix bugs, and update their operating parameters based on latest research. This update process is often *remotely over the Internet* in order to minimize the disruption to the user. However recently, instances have been found where the update process is not authenticated. This seems to be a larger trend with IoT systems that often seem to allow anyone to update their devices [36]. A case in point is the recent Food and Drug Administration recall of Hospira's infusion pumps because the received software and libraries during on-field updates in an unauthenticated manner allowing an adversary to man-in-the-middle the whole process [2] and compromise the device. Even though the infusion pump vulnerability was focused on an actuator medical system, the same applies to sensing medical devices as well.

Moreover, vulnerabilities also exist in the sensors themselves. For instance, sensors are susceptible to a whole class of sensory-channel threats that involve interfering with the transducers of the sensors and introducing arbitrary sensor measurements into the system again from *relative proximity*. This sensor hijacking can be performed using a variety of stimuli including electromagnetic induction [17, 27], light [41, 45, 47], and acoustic waves [24, 25]. Such sensory-channel attacks can not only be used to tamper with the sensor measurements [27], but also enable arbitrary code execution under specific conditions, as we ourselves recently identified [17].

Finally, physical compromise of the sensor directly or its replacement with a malicious device is yet another way to compromise the WMIoT and thus harm the user. Figure 1 lists the broad class of attacks that can lead to sensor hijacking on WMIoT.

1.2 Challenges in Detecting Sensor Hijacking

Detecting sensor-hijacking attacks on WMIoT is a challenging endeavor. This is because WMIoT is very different from traditional desktop and smartphone computing domain in several important ways. For example, in WMIoT user safety is given at

[1] In addition, attacks on pacemakers and insulin pumps also affect their actuation capabilities; however, we do not focus on such attacks in this work. That being said, in this work, we are detecting a more upstream attack, which will itself reduce the overall chances of incorrect actuation, assuming the actuation element is not directly attacked.

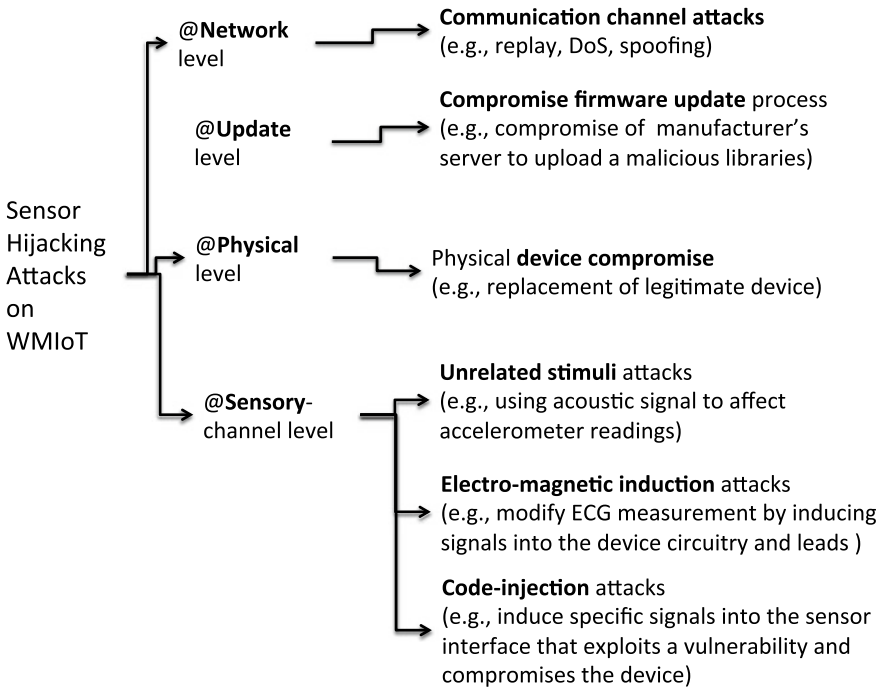


Fig. 1 A comprehensive classification of sensor-hijacking attack categories

the most importance. The devices in the WMIoT need rigorous device certification from agencies like the FDA. Furthermore, the medical devices themselves have limited computational capabilities that precludes effective implementation of security primitives. Traditional approaches to security that work well in the desktop and smartphone world are insufficient to deal with security problems in the WMIoT context. For instance:

- *Communication encryption-based solutions are insufficient.* The sensor-hijacking attacks compromise the source (sensors) of the data flow within the WMIoT. Consequently, no amount of securing the communication in the middle is therefore going to protect against user harm.
- *Malware detection solutions are not easy to introduce on medical devices.* Medical devices platforms are extremely heterogeneous and often built using very constrained embedded platforms. This makes deployment of general-purpose antivirus (AV) software infeasible without incurring extensive customization costs [22]. Further, even if AV software is present, in many cases manufacturers switch them off during firmware updates [22].
- *Hardware-oriented solutions are expensive.* Researchers have proposed upgrading the device hardware to prevent some of the hijacking attacks using techniques such as tamper-proof hardware, sensor shielding, or advanced filtering systems

[27]. These approaches increase the cost, weight, and energy consumption of the devices, thus affecting the deployment and wearability of the device along with the mobility of the user.

Given the various barriers that exist in making the medical devices more secure, we tackle the problem from a different angle. In our efforts, we are focused on developing an *attack-agnostic way* to secure the WMIoT systems against sensor-hijacking attacks. We do this by observing that the goal of any security solution with respect to WMIoT should be to ensure that the user is protected from harm. This means that the adversary should not be allowed to introduce incorrect user state information into the system such that wrong diagnosis and treatment are made. Therefore, if we can analyze the data being received from the sensors, we should be able to determine whether the data has been legitimately measured from the user or it has been tampered by someone. In this regard, our work has focused on detecting attacks on sensor data being output by electrocardiogram (ECG) sensors in WMIoT. The reason we focused on ECG sensors is because (1) ECG is one of the most important vital signs collected by a variety of WMIoT; (2) ECG is a representation of the cardiac process, which is very important to monitor for any user; and (3) ECG sensors have already been demonstrably compromised in variety of contexts [27, 30].

1.3 ECG Sensor Hijack Detection

In general, an ECG measurement has two key characteristics that can be manipulated by sensor-hijacking attacks such as *temporal* and *morphological* [40]. The manipulation of temporal characteristics involves modifying the timing information of the ECG complex (e.g., inter-beat-interval). This can result in situations where a user who has tachycardia (fast resting heart rate) being shown to have normal sinus rhythm ECG or vice versa. Similarly, manipulation of the morphological characteristics involves modifying the shape of the ECG complex that results in situations where a user's atrial fibrillation is misread as normal ECG or vice versa. Figure 2 shows examples of temporal and morphological changes in the ECG of the person and its visual manifestation.

One obvious way of addressing this problem is to deploy redundant ECG sensors on the user. This way even if one of the ECG sensors is compromised, we can detect that by correlating its output with that of other ECG sensors that may not be compromised. There are several issues with this approach: (1) If one ECG sensor is compromised, so can all the others; (2) it requires the deployment of a large number of sensors on user, which affects the usability of the WMIoT and therefore its effective in user care. Another approach to address this problem could be to have one ECG sensor but maintain a repository of historical ECG data form the user and correlate the current ECG measurements with historical observations. These solutions may work for healthy user whose ECG characteristics rarely change over time; however,

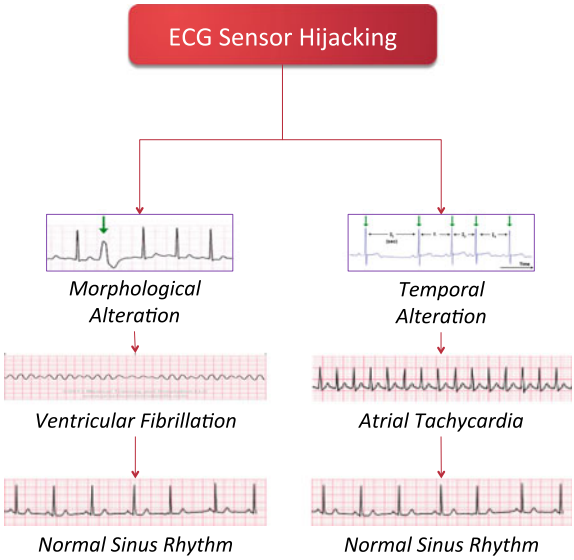


Fig. 2 Example of temporal and morphological alterations

it does not work for users who develop cardiac complications or generally for users with cardiac problems [18]. We therefore need an approach that does not require redundant ECG sensors nor depends on historical ECG data for its detection.

In this chapter, we present a detector for identifying sensor-hijacking-based temporal alteration of electrocardiogram (ECG) measurements in a WMIoT that relies on redundancy of signal characteristics rather than redundant devices. Our principal idea is to use another *reference signal* related to the ECG, also measurable from the user, as a way to identify if the ECG signal has been tampered with. The idea is to leverage the fact that different physiological signals generated by the same underlying physiological process are inherently correlated, i.e., they share similar features among them. Therefore, a surreptitious modification of one without the modification of the other can be identified with changes in features. For example, electrocardiogram and arterial blood pressure (ABP) are different manifestations of the cardiac process and the both signals track each other. Hence if we build a model that uses features that capture their behavior in tandem then a unilateral change in the ECG signal without a corresponding change in the ABP signal will be caught.

Of course our work relies on the assumption that the reference signal has not been tampered with. However, we believe that this is not a far-fetched assumption because detecting any sensor hijack requires a reference source that can be trusted to determine that sensor is indeed malfunctioning. If such a reference source is not present or if cannot be trusted, then no detection is possible. Further, there can be multiple possible reference signals that can be utilized. Therefore, ABP is only one option for ECG hijack detection, any cardiac signal can be used to detect ECG hijack. Furthermore, with this work we are suggesting that models based on characteristics of interrelated signals can be a powerful method for detecting hijacking and an

important alternative to device redundancy. A secure, untampered reference signal can be integrated in many ways in to the WMIoT: (1) It could be a part of a secure base station (e.g., Amulet platform [32]) that measures the reference signal purely for sanity checking the received ECG signal, (2) it could be a part of a suite of physiological monitoring sensors within the WMIoT, or (3) it could be a synthetic signal generated using a generative model based on historic measurements of the reference signal from the person (assuming the user's physiology has not changed substantially since then).

Consequently, to identify if the ECG sensor's measurement has been temporal altered, we train a *temporal alteration detection* model, at the base station that captures the commonalities of the ECG measurement with that of a correlated physiological signal, the arterial blood pressure (ABP) signal. Under normal situations, both ECG and ABP, which are measured synchronously, produce features that are not observable when the ECG signal is altered without a corresponding change in the user's physiology, thus indicating alteration. We used ABP as a reference signal mainly because of the availability of the dataset. We can use almost any other cardiac signal to perform the same kind of detection. That being said, ECG and ABP are often measured in tandem in a variety of situations [42, 46]. Numerous devices are available that measure ECG and continuous blood pressure in tandem as well [31, 46].

Note that we in [18] presented a preliminary version of the temporal alteration detector, where we used ABP and respiration signals as reference to determine ECG alteration. However, the version required over 60 min of ECG, ABP, and respiration data to train its model and 5 min of ECG, ABP, and respiration measurement before it could detect any attack. This was of course not optimal for WMIoT environments, where we want to be able to detect any attack on the sensors quickly. In this chapter, we present a new version of the detector that performs the same task in several orders of magnitude less time, only 1 min of ECG and ABP measurements for training the model and 5 s for detecting an attack, while maintaining comparable overall detection accuracy (around 97%).

1.4 Chapter Organization

The rest of the chapter is organized as follows. Section 2 presents the related work. Section 3 discusses the system and threat model along with the problem statement. Section 4 presents some background information about ECG, ABP, and relationship. Section 5 presents the model for detecting the temporal alteration of the ECG signal. Section 6 presents the evaluation of the detector. Section 7 presents a discussion of some of the limitations of the model presented. Finally, Sect. 8 presents the conclusions and future work.

2 Related Work

Not much work has been done in the domain of detecting sensor-hijacking attacks. In [27], the authors present several preventive solutions for sensor-hijacking attacks. However, these solutions require the sensor hardware to be upgraded through improved shielding and adaptive filtering techniques, which is hard to do without increasing the complexity and cost of the limited capability sensors. Therefore, we need a *detection solution* that executes at the base station (which typically has considerably more computational power) and can identify sensor-hijacking attacks through analysis of the measurements.

Work on detecting anomalous sensor measurements has largely focused on the benign case of fault detection. Fault detection in sensors in a WMIoT has involved the adaptation of sensor-redundancy-based methods from wireless sensor network domain to the WMIoTs [26, 28, 34, 43]. However, almost all the approaches are designed for motion and gait monitoring WMIoTs, and these kinds of WMIoTs naturally require considerable sensor redundancies (multiple sensors of the same type). In [37], the authors identify faults in a sensor by correlating its data with different sensors measuring related stimuli. Specifically, the paper focuses on detecting permanent faults in ECG signals based on ventricular pressure signals. Their approach builds a rule-table for various combinations of blood pressure and heart rates and determines whether the observed data fall within these expected bounds and, if not, then the sensors are deemed faulty.

An interesting approach for detecting medical device misbehavior has been developed by Kevin Fu's team at Michigan, called WattsupDoc [22]. The approach uses a supervised machine-learning model to learn a hospital-based medical device's (a drug compounder) behavior with respect to the amount of current it draws. However, such an approach would require device instrumentation in wearable systems, which may not be feasible because of the warranty requirements and limited technical expertise of the users who use and manage them. Consequently, in this project we plan to develop a solution that detects adversarial manipulation of device output without assuming: redundancy of devices, historical device output to be correlated with the current output, and device instrumentation or shielding.

As mentioned before in our previous work [18], we detect malicious temporal alterations of an ECG measurement using ABP and respiration measurements as references. The approach had a training time of 60 min (i.e., needed 60 min of physiological signal measurements to build the detector model) along with 5 min of ECG, ABP, and respiration measurements to detect (i.e., detection time) a sensor hijack, with is considerably slow for an application that is primarily used for real-time monitor of a person's health. In [19, 20], we demonstrate a version of the morphological alteration detector using ABP as reference, which is considerably accurate and much faster (training time of 20 min and detection time of 3 s) than our previous attempt at temporal change detection. In this chapter, we present an approach that remedies the situation for the temporal case.

3 System Model, Threat Model, and Problem Statement

System Model: We assume, the WMIoT consists of a number of wearable medical devices (i.e., *sensors*) (See in Fig. 3). These sensors are low-capability devices that collect physiological data from the user at regular intervals and forward that data to a highly capable sink entity, which we refer to as the *base station*, for processing and storage. The base station provides a root of trust for our system. The base station is assumed to be immune to attacks from the adversary. Examples of systems that are suitable to be base stations include the Amulet platform [32] developed at Dartmouth College. The Amulet system is designed to be energy-efficient, secure, and always present aggregator device with a watch-like form factor. Sensors in a WMIoT typically connect with the base station over a wireless network.

Threat model: The principal **goal of the adversaries** is *to compromise the safety of the user, through sensor-hijacking attacks.* In this regard, we will discuss some of the principal assumptions we make regarding the threats faced by our system model. We assume the adversaries possess the following characteristics. They can mount sensor hijacking on (a subset of) sensors through one of the many attacks listed in Fig. 1 (in Sect. 1) to misrepresent the current user state. In order to mount sensor-hijacking attacks effectively, the adversary needs to have some knowledge of the wearable medical system and the user. This resultant user medical state modification from the sensor-hijacking attack can take three forms: (1) by introducing arbitrary noise to the original sensor measurements; (2) by replaying historical sensor measurements stolen from the user in the past as current measurements; and (3) by replacing the actual sensor measurements with measurements belonging to another user.

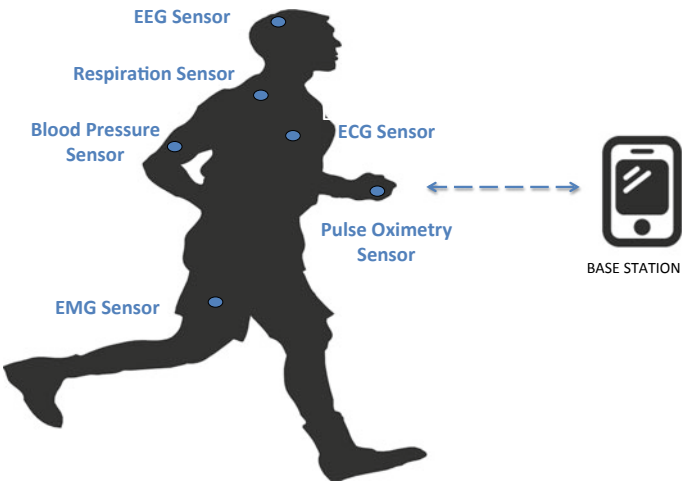


Fig. 3 Wearable medical internet of things (WMIoT) system

In the case of introducing arbitrary noise, the user and their caregivers will immediately be able to see the noise in the signal, which they can either ignore or investigate depending upon the larger context of the user's health. One can imagine a situation adversary deliberately introduces noise at critical parts of the measurement time-series and thus try to hide the emergence of specific medical conditions that need urgent action. However, this is unlikely because the adversary would have to introduce noise in real time to mask specific medical conditions, which is difficult to do with precision. Further we could have a policy in place which requires the caregiver to check on the user in the event of noisy output. The second form of hijacking attack where the adversary replays a historical ECG measurement would require adversaries to access to a user's past medical records, which we assume they do not have. *Consequently, in this chapter we focus on the third case, where actual ECG measurements are replaced with measurements belonging to another user.*

We *do not consider* eavesdropping or passive side-channel attacks per se, as these attacks by themselves do not affect user safety. However, eavesdropping and passive side-channel attacks can be used as a precursor to attacks on the sensor itself. In such cases, we detect the eventual sensor-hijacking attacks they enable. As a final note, as we are dealing with relatively closed systems, we assume that the adversary cannot poison the training data used by the machine-learning models, which form the basis of the detector.

Problem Statement: Our goal is to develop a detector for identifying temporal alteration of electrocardiogram (ECG) measurements in a WMIoT using the synchronously obtained measurements of a *trustworthy reference*, the arterial blood pressure (ABP) measurements.

4 Background

Before we delve into the details of ECG alteration detection based on correlated signal features, we provide some background information on the principal signals we consider for this work, i.e., electrocardiogram (ECG) and arterial blood pressure (ABP) and the interrelationship between them. This is important because we leverage such interrelationships for detection ECG alteration for sensor-hijacking attacks.

ECG is the measurement of the electrical representation of the cardiac process of a person. As shown in Fig. 4, a single ECG complex consists of P, Q, R, S, and T waves. The P-wave is observed during atrial depolarization (which causes the blood to be pushed to the ventricles), the QRS complex is observed during the rapid depolarization of the right and left ventricles (which causes the blood to be pushed out of the ventricles into the lungs and the rest of the body), and the T-wave is the depolarization of the ventricles. The time difference between two R-peaks is known as an *RR-interval*. The RR-interval refers to the beat-to-beat variations in heart rate and is a measure of heart rate.

Atrial blood pressure (ABP), on the other hand, is the continuous measurement of blood pressure and can be measured non-invasively [7] much like ECG. As shown

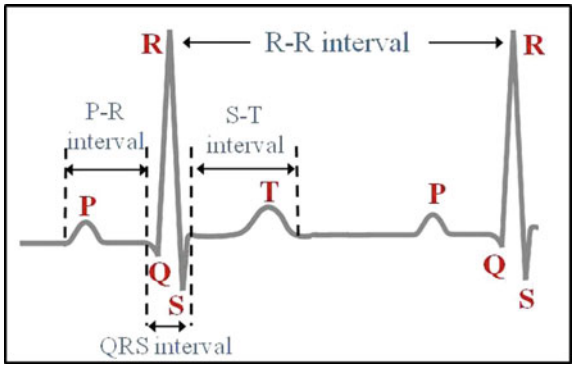


Fig. 4 P, Q, R, S, T waves in a typical ECG

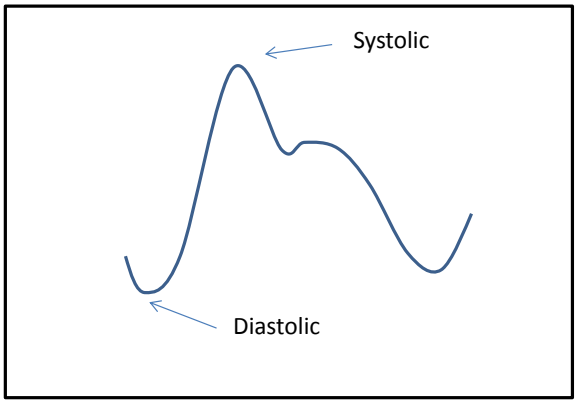


Fig. 5 Systolic peak and diastolic trough in ABP

in Fig. 5, a typical atrial blood pressure contains the trough representing the diastolic blood pressure and a peak which is the systolic blood pressure. Diastolic troughs occur near the beginning of the cardiac cycle and systolic peaks occur when the ventricles contract. As ECG and ABP signals are both measures of the cardiac process (see Fig. 6), they track each other. For example, an R-peak in the ECG signal will typically be followed by a systolic peak in the ABP signal as both represent the compression of the ventricles that results in the blood being circulated through the entire body via the Aorta (see Fig. 6). Therefore, under normal conditions, the RR-intervals and systolic peak intervals in the ABP signal (referred to as *SS-intervals* from now on) are highly correlated, as they are measures of the same phenomenon (ventricular compression) [38]. Similarly, pathologies in the cardiac process that results in abnormal ECG wave form are also reflected in the ABP signal [1].

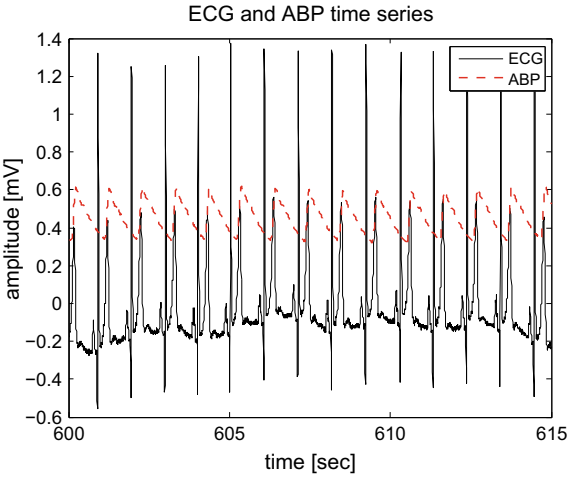


Fig. 6 ECG and ABP signals in time domain

5 Detecting Temporal ECG Alterations

In this section, we present a new version for detecting temporal alteration of electrocardiogram (ECG) sensor measurements in a WMIoT that is a order of magnitude faster than our previous attempt at the problem [18] while maintaining high accuracy rate. The overall approach used by our detector is the same as in our previous work [18]. It works by training a *user-specific* supervised-learning model that includes features that capture the interrelationship among synchronously measured ECG and another signal(s), which in this case is arterial blood pressure (ABP) from a particular user. The model also includes features collected from ABP measured from the user and ECG measured from several different users (thus modeling the attack where a user's ECG is replaced with someone else's as part of the sensor hijacking). Once the model has been trained, we then use features extracted from snippets of synchronously measured ECG and ABP signals from the user and feed it into the model. The model generates an alert if it determines that the signals came from two different users. Figure 7 shows our system setup. It contains three main steps: (1) extracting features that capture the interrelationship among ECG and ABP, (2) training a user-specific model, and (3) detecting altered ECG measurements based on the newly received ECG and ABP snippets. We generate a *user-specific model* to capture the features of the user's cardiac process as observed through the ECG and ABP signals to get a baseline for expected behavior and unexpected behavior for our detection system. We now describe each of the three stages below.

Feature Extraction: We view alteration of ECG measurements, with the intention of providing incorrect data about the user, to manifest itself as *temporal changes* in the output ECG signal. Temporal changes are associated with the interval between two consecutive R-peaks being misreported. Here, as we are focused on temporal properties of ECG signals, we first transform the ECG signal into a series of

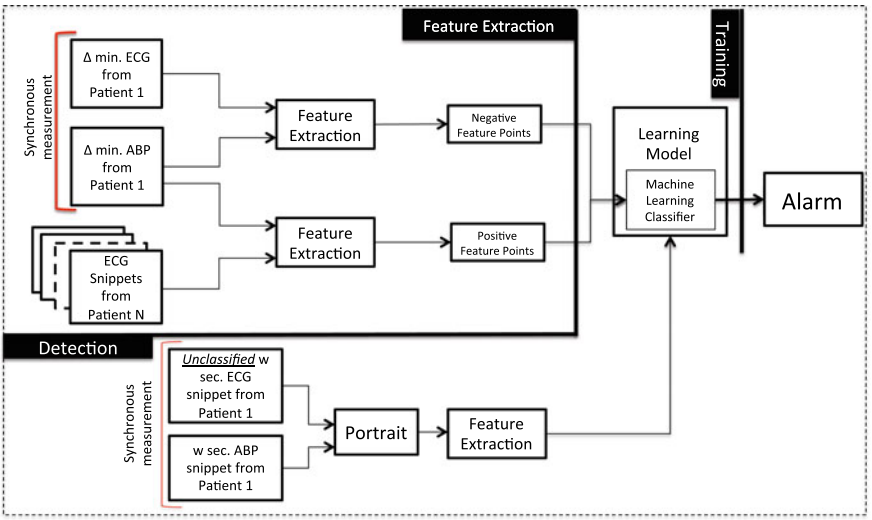


Fig. 7 Detection of temporal alteration of ECG measurements

inter-beat-intervals by detecting the R-peaks and calculating the time difference between two consecutive R-peaks. (Note that before extracting features, the signals cleaned, to eliminate noise such as baseline wandering.) The RR-tachogram thus produced forms our *candidate signal*. We then correlate this candidate signal with a *reference* ABP signal and extract feature points in tandem. These feature points will then be used to train a user-specific model and used to detect ECG alterations. In all, we extracted a set of 13 features from candidate and reference signals (see Table 1). These features can be classified into two categories: (1) *Time-Domain Features*, which include (i) average and standard deviation of the time elapsed between two consecutive R-peaks in the ECG signal, (ii) average and standard deviation of the time elapsed between two consecutive systolic (S) peaks in the ABP signal, (iii) average amplitude of the systolic peak and diastolic peak, (iv) average, standard deviation, and root-mean-squared of the pulse transit time (PTT) and backward PPT. (2) *Frequency-Domain Features*, which include area under the curve of the magnitude squared coherence (MSC) curve computed between ECG and ABP signals.

Above, the pulse transmit time (PTT) is defined as the distance between R-peak and the systolic peak which is preceded by that R peak (shown in Fig. 8). Similarly, backward PTT is defined as the distance between R-peak and the nearest systolic peak shows ahead of that R-peak (shown in Fig. 8). Further, magnitude squared coherence (MSC) is the measure of spectral coherence and measures the causality between the two signals. The MSC of two signals, signal $x(t)$ and signal $y(t)$, is defined as follows: $C_{xy}(f) = \frac{|P_{xy}(f)|^2}{P_{xx}(f) * P_{yy}(f)}$, where $P_{xx}(f)$ and $P_{yy}(f)$ denotes the power spectral densities of signal $x(t)$ and signal $y(t)$, respectively, and $P_{xy}(f)$ denotes the cross power spectral density of these two signals. We use MSC as feature for our

Table 1 Feature summary for ECG temporal alteration detection

Type	Feature
Time domain	Average RR-interval
	Standard deviation of RR-interval
	Average SS-interval
	Standard deviation of SS-interval
	Average SBP amplitude
	Average DBP amplitude
	Average pulse transmit time (PTT)
	Standard deviation of PTT
	Root mean square of PTT
	Average backward PTT
	Standard deviation of backward PTT
	Root mean square of backward PTT
Frequency domain	Area under MSC curve of ECG and ABP

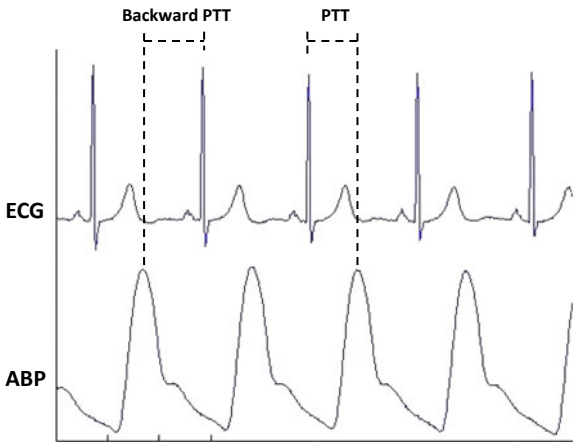


Fig. 8 PTT and backward PTT

work because it has been shown that it provides an important measure of the power interchanged between the signals in determined frequency bands (around 0.1 Hz and around the respiration frequency) [15].

Training: In order to account for the individual variation in the physiological processes, we build a *user-specific* model for each user on whom we tested our system. We tried several classifiers in this regard. However, we include results of the best performing classifier, namely Support Vector Machine (SVM). To train these classifiers, we first extract the aforementioned 13-dimensional features from Δ_t minutes of synchronously measured ECG and ABP signals from the same user and label these as *negative class* points. The feature extraction is done using sliding window of size $w_t < \Delta_t$, which is moved over the two synchronously measured signals. Each

w_t-sized window of data thus produces one feature point for the system. We then extract the aforementioned features using snippets from ECG signals with ABP signal from different users and label these as *positive class* points. Once the negative and positive points are collected, we feed them into the SVM classifier to generate a user-specific model. The use of SVM presents advantages in that it is well understood, relatively easy to understand, and has excellent tool support, and can be implemented cheaply on resource-constrained base stations [21].

Generating Alerts: After model training stage is completed, we can use the trained model for a user to decide if any newly received snippet of ECG and ABP signal has been temporally altered or not. Again, we use feature generation method for w_t-sized long synchronously measured ECG and ABP snippets to generate a feature point, and then feed this feature point to our user-specific model. Then the user-specific model will output a label for this feature point as negative or positive. If the point is deemed positive, we raise an alarm.

6 Evaluation of ECG Temporal Alteration Detector

In this section, we evaluate performance of this new version of ECG alteration detection model. Our goal with the validation is to demonstrate the ability of our models to detect alterations in the temporal properties of ECG signals induced by an adversary while minimizing the latency in training the models and attack detection time.

We use the following **metrics** to evaluate the ability of our detector to detect alterations of ECG sensor measurement: false positive rate, false negative rate, and balanced accuracy. We define *false positive* (FP) as the fraction of the cases where an unaltered ECG sensor measurement is misclassified as altered. Similarly, *false negative* (FN) as the fraction of the cases where an altered ECG sensor measurement is misclassified as unaltered. We define *balanced accuracy rate* (BAC) as the sum of the half of the true negative rate (fraction of the unaltered ECG sensor measurement was classified as unaltered) and half of the true positive rate (fraction of the altered ECG sensor measurement was classified as altered). We use the BAC because of the metric weighs the positive and negative samples equally. This is important given that we have an imbalanced sample with many more positive examples than negative ones during the learning phase [48]. In addition, we also track the training time (Δ_t), that is the amount of time for which we need to measure ECG and ABP signals such that we can build an accurate detector, and detection time (w_t), which determines how quickly our model detect an attack. Even though we compute these metrics for each user in our dataset, we present summary statistics of these metrics over all users.

Dataset: The first step in validating the ECG temporal alteration detection model is to train a user-specific learning model. We collected data belonging to 28 users from the MIT PhysioBank Fantasia and MGH databases [29]. We chose these particular users from these databases as they contain both ECG and ABP signals. Furthermore, the Fantasia database is made up of healthy users, while the MGH database mainly

Table 2 Balanced accuracy rate for different w_t with fixed $\Delta_t = 41$ min

Window size w_m (s)	Avg. FP (%)	Avg. FN (%)	Avg. BAC (%)
5	1.27	3.28	97.72
15	1.33	0.54	99.06
30	2.65	0.03	98.66
60	2.35	0.03	98.81

contains data from users with specific cardiac conditions (i.e., ailment). We searched the MGH database to specifically choose users whose cardiac condition manifested itself in temporal variation in the measured ECG signal. We categorize the list of 28 users into three user types based on their ECG signals: (1) *Normal* user type indicates users who did not suffer from any cardiac conditions and had normal sinus rhythm ECG. Our dataset had 13 Normal users, 6 males, 7 females, with an average age of 44.46 (std 25.52). (2) *Abnormal T* user type indicates users with consistent tachycardia (fast resting heart rate) or bradycardia (slow resting heart rate). Our dataset had 6 Abnormal users, 4 males, 2 females, with an average age of 61.4 (std 19.25). (3) *Mixed* user type indicates users whose ECG signal showed both normal as well as tachycardia or bradycardia rhythms. Our dataset had 9 Mixed users, 5 males, 4 females, with an average age of 44.78 (std. 20.39).

Parameter Selection: Given the dataset we now extract features and train a user-specific model using the data collected from the databases. Two important questions to address at this point are the window size (i.e., w_t) we need based on which one 13-dimensional feature point can be extracted, and the duration of ECG and ABP data (i.e., Δ_t) we need for the training phase. Note that as we need a window size amount of ECG and ABP measurements for generating one feature point, the window size is the same as the *detection time* metric for our system.

To test which window size w_t works best, we set Δ_t to be a fixed value 41 min (as that was the maximum duration of ECG and ABP measurements that were available for the users in our dataset) and tested our temporal alteration detector with different values w_t as 5 ,10, 30, and 60 s. To generate the negative feature points, we used 41 min synchronously measured ECG and ABP signals from the same user. To generate the positive feature points, we used each user's ABP with a randomly selected ECG snippet from every other user in the dataset. These feature points are then fed into a machine-learning classifier for training purposes. We used Support Vector Machine (SVM) and used tenfold cross validation to test the detector built. Table 2 shows the metrics BAC, FP, and FN for temporal alteration detector using SVM with different window size w_t. We can see that the temporal alteration detector works best with a window size of 60 s. Further, we can see that, for these four different window size w_t, the temporal alteration detector all achieve at a really high BAC rate with an accuracy difference between 5 and 60 s around 1%. Therefore, in the rest of the experiment, we set $w_t = 5$ s, since the window size w_t determines how fast and frequently the

Table 3 Balanced accuracy rate, FP, and FN for different Δ_t with fixed $w_t = 5$ s

Training data (min) Δ_m	Avg. FP (%)	Avg. FN (%)	Avg. BAC (%)
1	2.98	3.77	96.63
5	1.79	2.29	97.96
10	1.96	2.19	97.92
15	2.12	2.04	97.92
20	2.23	2.19	97.79

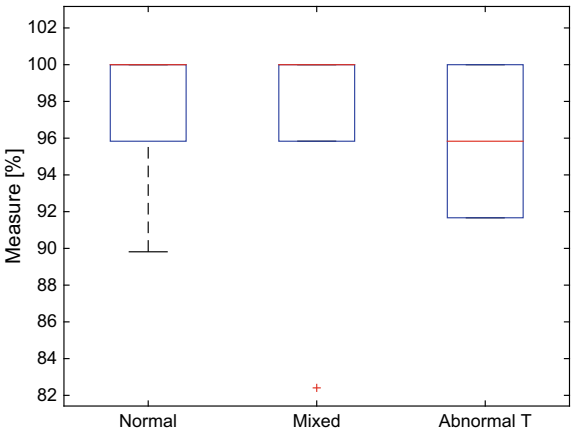


Fig. 9 Balanced accuracy rate for ECG temporal alteration detector

temporal detector outputs the decision whether the ECG signal is temporally altered or not.

We then determine how the performance of our ECG temporal alteration detector is affected by our choice of the size of training data. In this regard, we set the window size to 5 s and our model with for different amounts of ECG and ABP (i.e., Δ_t) training data. Using each Δ_t, we generated a set of negative and positive feature points, and used SVM classifier to train the detector, respectively. We also used tenfold cross validation to test the model built. Table 3 shows the metrics BAC, FP, and FN for the temporal alteration detector with different Δ_t. We see that the temporal alteration detector works best with 5 min of training data. However, we chose $\Delta_t = 1$ min because it faster to train and also the difference in accuracy from between 5 and 1 min is only 1.3%. In the rest of the experiments, we set training data size $\Delta_t = 1$ min.

Detection Results: Figs. 9 and 10 show the box plots for BAC, FP, and FN rates of ECG temporal alteration detector using Support Vector Machine classifier based on the different groups of users ($\Delta_t = 1$ min, $w_t = 5$ s). In terms of detection accuracy, we can see that even though the Abnormal T user group has a considerably higher spread comparing with Normal user group, the balanced accuracy of this group is

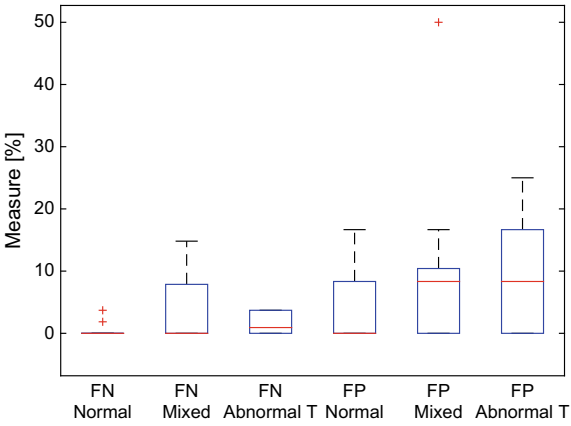


Fig. 10 False positive rate and false negative rate for ECG temporal alteration detector

still high. If we look at the false negative rate, all three groups work pretty much similar and the rate is close to zero. The false positive rates are low as well for the normal case in the median case, but we can see a few outliers where the rates much higher. In the case of the mixed users, the false positives rates showed similar trend to what we saw with the normal users. The worst performing set was the abnormal set, where even the median false positive rate was close to 8%. Given the larger inter-quartile range for the false positive rate for the abnormal case of about 15%, it is clear that there is a lot of variability in the results.

Comparison with Previous Work: In our previous work in [18], we developed a different version of temporal alteration detector that used ABP and respiration signals as reference. Table 4 summarizes the results, including the average BAC, FP, and FN of the detector in this chapter (current version) with our previous version in [18], and the base case that analyzes historical inter-beat-interval pattern (RR-intervals), which we call the *RR-only* cases. In terms of detection accuracy and error, we can see that RR-only is reasonably accurate (average BAC of ∼87.41%) but has large false positives and negatives particularly for the Mixed dataset where users exhibit both healthy and unhealthy ECG signals. Both the previous version and current version of the detector consistently outperform this base version for all user types not only in terms of accuracy but also false positive and false negative rates. In comparison to the previous version of the detector, our current detector shows less than 1% decrease in accuracy. However, the biggest advantage for our the work presented in this chapter is that it is very fast to train its underlying model and (1 min as opposed to 60 min) and fast to detect the attacks (5 s as opposed to 5 min), which is a crucial property for actual deployment.

Table 4 Performance Summary

Method	Avg. BAC (%)	Avg. FP (%)	Avg. FN (%)	Training Time (s)	Detection Time (s)
Current version	96.63	2.98	3.77	60	5
Previous version in [18]	98.46	2.44	0.65	3.6 K	300
Base version (RR-only)	87.41	16.83	8.36	3.6 K	300

7 Discussion

In this section, we discuss the limitation of our detector and the results we presented in the previous section.

- **Combining Temporal and Morphological Detectors**: In our previous work, we have developed a morphological alteration detection model that uses a window size is 3 s and the training time is 20 min [20]. In light of the approach presented in this chapter, we now have two separate models which together can detect any type of ECG alteration. Therefore, to identify if any newly received snippet of ECG measurements have been maliciously altered or not, these two models have to run in parallel, and the final decision has to be made based on both output of the two models. However, combining the outputs of the two models to detect ECG alteration is not straightforward because the datasets that used to evaluate these two models are different, we are focused on the temporal alteration of ECG measurements, while [20] we were focused on the morphological alteration of ECG measurements. Moreover, the window size for the temporal alteration detection model (1 s) and the window size for the morphological alteration model (3 s) are not the same. This leads us to the question, how to combine the output of two models that are based on the different length of the ECG measurement, and make a final decision. It cannot obviously be done synchronously. Since the difference between the window sizes is so small we could simple announce temporal alterations every-second, while we announce morphological changes every 3 s. The bigger issue occurs during training or retraining of the models, which show a marked difference between the two detectors. Every time we take the detectors offline (to retrain them) we have to wait for 20 min before redeploying them, even though the temporal detector is ready is fairly short amount of time. This means despite having a short training time, temporal detectors when used as part of the larger alteration detection infrastructure cannot benefit from this property. We are currently working on building a unified model that can detect both temporal and morphological alterations of ECG measurements.
- **Dealing with Variations in ECG Signal Behavior**: The development of the machine-learning model for the aforementioned detector is non-trivial because

physiological characteristics display a large variety of normal behaviors for a given user and between users. This means the machine-learning model not only has to be user-specific, but also has to capture a variety of transient idiosyncrasies of the user's physiology (e.g., ectopic beats) in order to be effective. In [33], the authors show that a good strategy in using machine learning to detect adversary-induced deviations from norm, is to use a family of detectors. This allows the detectors to compensate for the population drift, where the patient's physiological signals change their characteristics over time. We therefore propose to deploy a family of detectors, each with its own machine-learning model, to detect sensor-hijacking attacks and compensate for the uniqueness of the patient's physiology. Figure 11 illustrates an overview of our detection system, which employs a family of detectors. After initial preprocessing where we clean the candidate and reference signals (e.g., de-noise it), we use them as input to a family of detectors, each of which has its own transformation function, feature extractor, and machine-learning model. Each detector is designed to capture user-specific physiological variations. A decision fusion module then takes results of the individual detectors and produces a final decision (and an alarm).

- **Dealing with Drift in Physiological Signals Over Time**: Our approach relies on the interrelationship among ECG, and ABP signals to operate. If a user's physiology changes over time, the models have to adapt as well. In our current design, both two models are trained in an offline fashion with only the alert generation happening online. This means that the model has to be retrained every so often in order to capture the current state of the user's health. One approach is to automate the re-learning based on a schedule. However, choosing the inter-relearning interval has to be done carefully. Too short an interval would lead to unnecessary re-learning and too long an interval would result in increased errors. Determining the optimal model retraining frequency for our work is probably a user-dependent parameter. For relatively healthy users, the retraining need not happen often, while for individual cardiac conditions, the training has to be done more frequently depending upon the actual condition, how acute it is, and any medications they might be

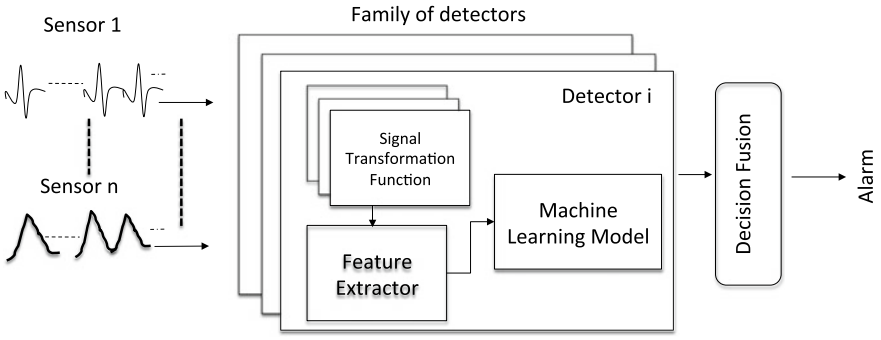


Fig. 11 Sensor-hijacking attack detection system

taking. The calculation of optimal model retraining is a non-trivial problem in its own right something we plan to explore in the future.

- **Using Improved Machine-Learning Models**: In this chapter, we used supervised learning to train the user-specific model to illustrate our ideas. Supervised learning is based on the training dataset as the ground truth. The positive class points are generated by using ECG from a group of arbitrary users. If the ECG is altered by using an ECG snippet that is collected from a new user who is outside that group, then the detection system may not work as well. Thus, to solve this problem, the possible solution might be: (i) use unsupervised learning instead of supervised learning; (ii) enrich the dataset thus the altered ECG could be representative; (iii) build a general model, and then using transfer learning to build a user-specific model. We plan to explore this in the future.

8 Conclusions

In this chapter, we presented a novel version of our sensor hijacking-based temporal alteration detector for ECG measurements in a Wearable Medical Internet of Things (WMIoT) context. This version leveraged the similarity of ECG to another physiological signal that measures the cardiac process, arterial blood pressure signal to detect the alterations. We built a new detector for temporal alterations that is much faster to train its underlying model and then detect alterations as compared to our previous work in [18]. Analysis of our detector demonstrated promising results with ∼97% accuracy in detecting ECG alterations for both healthy and unhealthy users with an order of magnitude less time for training and attack detection that the previous version.

8.1 Future Work

In the future, we plan to extend this work in several directions.

- We want to build a combined detector for both temporal and morphological alterations of ECG measurements are that synchronizes the amount of time it needs to train and then detect potential sensor-hijacking-based alterations.
- We want to implement this new temporal detector on an actual base station platform such as the amulet system [32] and evaluate the performance and computational cost. We have already done so for the morphological detector on amulet and the results were published in [21].
- Detection of an attack is only the beginning. Once the attack has been detected, it is important to alert the user of the attack and help them take mitigative actions to reduce the consequences of the attack. How to design such an alert system is an open problem.

References

1. Abnormal EKGs and Corresponding Arterial Waveforms. http://www.dynapulse.com/educator/WebCurriculum/Chapter%203/Abnormal%20EKG%20and%20Waveform.htm
2. Advisory (ICSA-15-090-03), Hospira MedNet Vulnerabilities. https://ics-cert.us-cert.gov/advisories/ICSA-15-090-03
3. AiQ. http://aiqsmartclothing.com/
4. Apple Watch. https://www.apple.com/watch/
5. Avery Dennison: Metria. http://www.averydennison.com/en/home/technologies/creative-showcase/metria-wearable-sensor.html
6. CardioMEMS. http://www.sjm.com/cardiomems
7. The clearsight system. http://www.edwards.com/eu/products/mininvasive/pages/clearsightsystem.aspx
8. Empatica. http://www.empatica.com
9. Empatica E4 Writstband. https://www.empatica.com/e4-wristband/
10. Fitbit. http://www.fitbit.com
11. HIPAA security and privacy rule. https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html
12. Jawbone. http://www.jawbone.com
13. Pancreum: The Wearable Artificial Pancreas Company. http://pancreum.com/
14. Wearable Biosensor for Diabetics: Nanotech News. http://anewdomain.net/2012/10/01/diabetes-patients-biosensor/
15. Baselli, G., Cerutti, S., Civardi, S., Liberati, D., Lombardi, F., Malliani, A., Pagani, M.: Spectral and cross-spectral analysis of heart rate and arterial blood pressure variability signals. Comput. Biomed. Res. **19**(6), 520–534 (1986)
16. BioHarness BT: http://www.zephyr-technology.com/wp-content/uploads/2012/01/ZEPHYR-GOES-STRAPLESS-AT-2012-CES.pdf
17. Brown, N., Patel, N., Plenefisch, P., Moghimi, A., Eisenbarth, T., Shue, C., Venkatasubramanian, K.K.: Scream: Sensory channel remote execution attack methods. In: Usenix Security Symposium (2016)
18. Cai, H., Venkatasubramanian, K.K.: Detecting malicious temporal alterations of ECG signals in body sensor networks. In: Network and System Security, pp. 531–539. Springer (2015)
19. Cai, H., Venkatasubramanian, K.K.: Poster: Detecting malicious morphological alterations of ECG signals in body sensor networks. In: ACM/IEEE International Conference on Information Processing in Sensor Networking (2015)
20. Cai, H., Venkatasubramanian, K.K.: Detecting signal injection attack-based morphological alterations of ecg measurements. In: 2016 International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 127–135. IEEE (2016)
21. Cai, H., Yun, T., Hester, J., Venkatasubramanian, K.K.: Deploying data-driven security solutions on resource-constrained wearable iot systems. In: 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), pp. 199–204 (2017)
22. Clark, S.S., Ransford, B., Rahmati, A., Guineau, S., Sorber, J., Xu, W., Fu, K.: WattsUpDoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices. In: USENIX Workshop on Health Information Technologies (2013). https://spqr.eecs.umich.edu/papers/clark-healthtech13.pdf
23. Dan Goodlin: Insulin pump hack delivers fatal dosage over the air. (2011). http://www.theregister.co.uk/2011/10/27/fatal_insulin_pump_attack/
24. Dean, R.N., Castro, S.T., Flowers, G.T., Roth, G., Ahmed, A., Hodel, A.S., Grantham, B.E., Bittle, D.A., Brunsch, J.P.: A characterization of the performance of a mems gyroscope in acoustically harsh environments. IEEE Trans. Ind. Electron. **58**(7), 2591–2596 (2011)
25. Dean, R.N., Flowers, G.T., Hodel, A.S., Roth, G., Castro, S., Zhou, R., Moreira, A., Ahmed, A., Rifki, R., Grantham, B.E., Bittle, D., Brunsch, J.: On the degradation of mems gyroscope performance in the presence of high power acoustic noise. In: 2007 IEEE International Symposium on Industrial Electronics, pp. 1435–1440 (2007)

26. Duk-Jin, K., Prabhakaran, B.: Motion fault detection and isolation in body sensor networks. Pervasive Mobile Comput. **7**(6), 727–745 (2011)
27. Foo Kune, D., Backes, J., Clark, S.S., Kramer, D.B., Reynolds, M.R., Fu, K., Kim, Y., Xu, W.: Ghost Talk: Mitigating EMI signal injection attacks against analog sensors. In: Proceedings of the 34th Annual IEEE Symposium on Security and Privacy (2013). https://spqr.eecs.umich.edu/papers/fookune-emi-oakland13.pdf
28. Galzarano, S., Fortino, G., Liotta, A.: Embedded self-healing layer for detecting and recovering sensor faults in body sensor networks. In: 2012 IEEE International Conference on Systems, Man, and Cybernetics, pp. 2377–2382 (2012)
29. Goldberger, A.L., Amaral, L.A.N., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.K., Stanley, H.E.: Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. Circulation **101**(23), e215–e220 (2000)
30. Halperin, D., Kohno, T., Heydt-Benjamin, T., Fu, K., Maisel, W.: Security and privacy for implantable medical devices. IEEE Pervasive Comput. **7**(1), 30–39 (2008)
31. Helo L.X.: http://helosmartwristband.com/helo-lx/
32. Hester, J., Peters, T., Yun, T., Peterson, R., Skinner, J., Golla, B., Storer, K., Hearndon, S., Freeman, K., Lord, S., Halter, R., Kotz, D., Sorber, J.: Amulet: An energy-efficient, multi-application wearable platform. In: Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM, SenSys '16, pp. 216–229 (2016)
33. Kantchelian, A., Afroz, S., Huang, L., Islam, A.C., Miller, B., Tschantz, M.C., Greenstadt, R., Joseph, A.D., Tygar, J.D.: Approaches to adversarial drift. In: Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security, AISec '13, pp. 99–110 (2013)
34. Kim, D.J., Suk, M.H., Prabhakaran, B.: Fault detection and isolation in motion monitoring system. In: 2012 Annual International Conference of the IEEE on Engineering in Medicine and Biology Society (EMBC), pp. 5234–5237. IEEE (2012)
35. Li, C., Raghunathan, A., Jha, N.: Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system. In: 13th IEEE International Conference on e-Health Networking Applications and Services (Healthcom), pp. 150 –156 (2011)
36. Lucian Constantin: Authentication bypass bug exposes Foscam webcams to unauthorized access. http://www.pcworld.com/article/2091180/authentication-bypass-bug-exposes-foscam-webcams-to-unauthorized-access.html
37. Mahapatro, A., Khilar, P.M.: Fault diagnosis in body sensor networks. Int. J. Comput. Inf. Syst. Ind. Manag. Appl. **5**, 252–259 (2013)
38. Malik, M., Bigger, J.T., Camm, A.J., Kleiger, R.E., Malliani, A., Moss, A.J., Schwartz, P.J.: Heart rate variability standards of measurement, physiological interpretation, and clinical use. Eur. Heart J. **17**(3), 354–381 (1996)
39. Mather, M.: Fact sheet: Aging in the United States. (2016). http://www.prb.org/Publications/Media-Guides/2016/aging-unitedstates-fact-sheet.aspx
40. McSharry, P.E., Clifford, G.D., Tarassenko, L., Smith, L.A.: A dynamical model for generating synthetic electrocardiogram signals. IEEE Biomed. Eng. Trans. **50**(3), 289–294 (2003)
41. Park, Y., Son, Y., Shin, H., Kim, D., Kim, Y.: This ain't your dose: Sensor spoofing attack on medical infusion pump. In: 10th USENIX Workshop on Offensive Technologies (WOOT 16). USENIX Association, Austin, TX (2016). https://www.usenix.org/conference/woot16/workshop-program/presentation/park
42. Ribeiro, D.M.D., Colunas, M.F.M., Marques, F.A.F., Fernandes, J.M., Cunha, J.P.S.: A real time, wearable ecg and continous blood pressure monitoring system for first responders. In: 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 6894–6898 (2011)
43. Sagha, H., del R Millan, J., Chavarriaga, R.: Detecting and rectifying anomalies in body sensor networks. In: 2011 International Conference on Body Sensor Networks, pp. 162–167 (2011)
44. Saria, S.: A $3 trillion challenge to computational scientists: transforming healthcare delivery. IEEE Intell. Syst. **29**(4), 82–87 (2014)

45. Shin, H., Son, Y., Park, Y., Kwon, Y., Kim, Y.: Sampling race: bypassing timing-based analog active sensor spoofing detection on analog-digital systems. In: 10th USENIX Workshop on Offensive Technologies (WOOT 16). USENIX Association, Austin, TX (2016). https://www.usenix.org/conference/woot16/workshop-program/presentation/shin
46. Sotera Wireless: http://www.soterawireless.com/visi-mobile/
47. Uluagac, A., Subramanian, V., Beyah, R.: Sensory channel threats to cyber physical systems: A wake-up call. In: 2014 IEEE Conference on Communications and Network Security (CNS), pp. 301–309 (2014)
48. Velez, D.R., White, B.C., Motsinger, A.A., Bush, W.S., Ritchie, M.D., Williams, S.M., Moore, J.H.: A balanced accuracy function for epistasis modeling in imbalanced datasets using multifactor dimensionality reduction. Genet. epidemiol. **31**(4), 306–315 (2007)

Cryptography in WSNs

Thomas M. Chen, Jorge Blasco and Harsh Kupwade Patil

Abstract Cryptography is the field for mathematically transforming messages to protect their confidentiality from eavesdropping. It is an essential foundation for all secure protocols. This chapter is a concise review of cryptography used in wireless sensor networks (WSNs) beginning with background on symmetric key encryption and the Diffie–Hellman key agreement protocol for sharing secret keys. Asymmetric key encryption and its usefulness for digital signatures and digital certificates are described. Finally, the unique constraints on the use of cryptography in wireless sensor networks are reviewed. A short survey of the literature is presented, which mainly deal with evaluating ciphers (particularly elliptic curve cryptography), proposing new approaches for key management, and implementing identity-based cryptography.

1 Introduction

Although easy to deploy, wireless networks are obviously vulnerable to eavesdropping by any radio receiver within a proximity close enough to pick up the signal. Therefore, wireless communications are routinely encrypted for confidentiality. Encryption guarantees a measure of privacy by mathematically transforming data messages (plaintext) into ciphertext that is not understandable by an eavesdropper without knowledge of the secret key to properly decrypt the ciphertext back to plaintext. If a strong encryption algorithm such as the standard Advanced Encryption

T. M. Chen (✉)
University of London, London, UK
e-mail: tom.chen.1@city.ac.uk

J. Blasco
Royal Holloway, University of London, Egham, UK
e-mail: jorge.blascoalis@rhul.ac.uk

H. Kupwade Patil
LGE Mobile Research, Santa Clara, CA, USA
e-mail: harsh.patil@lge.com

H. M. Ammari (ed.), *Mission-Oriented Sensor Networks and Systems: Art and Science*, Studies in Systems, Decision and Control 163,
https://doi.org/10.1007/978-3-319-91146-5_21

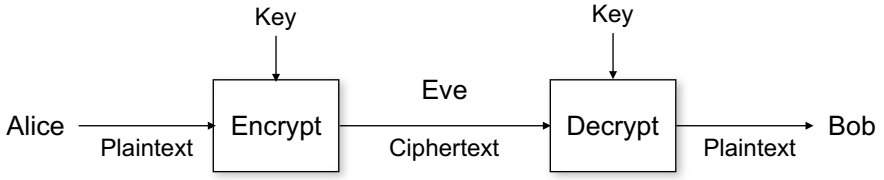


Fig. 1 A general cryptosystem

Standard (AES) [62] is used, the ciphertext should appear to be "scrambled" random without any apparent statistical structure.

Cryptography provides the mathematical foundations for the construction of secure protocols. It offers not only assurance of confidentiality but also means to authenticate identities and verification of data integrity. Although historically developed mainly for military or political applications, cryptography has become essential for all modern secure communications ranging from private conversations on mobile phones to electronic commerce in the Internet. Just as importantly, cryptography protects electronically stored data (e.g., on hard drives or in the cloud) from unauthorized reading.

A general cryptosystem consists of encryption and decryption processes, as shown in Fig. 1. It has become common practice to refer to the communicating parties as Alice and Bob; adversaries include an eavesdropper on the channel, Eve. The readable plaintext message M is encrypted into ciphertext $C = E(M)$ before transmission. The encryption must be reversible by decryption for Bob to recover $M = D(C)$. Encryption and decryption depend on keys that are the same in symmetric key cryptography but different in asymmetric key cryptography.

It may be tempting to hide all details of the cryptosystem in an effort to increase security, then Eve would have to discover the encryption algorithm (or cipher) as well as the key. However, this "security through obscurity" approach is dubious. First, it is often practically difficult to keep everything about a cryptosystem secret. Second, if an adversary manages to discover the encryption algorithm, the system may become seriously compromised (particularly if the encryption algorithm is weak). Third, undisclosed encryption algorithms are more likely to have vulnerabilities that make the system easy to crack. Standardized encryption algorithms have undergone extensive public review to verify that they do not have serious weaknesses. For these reasons, security through obscurity can give a false sense of security. Alice and Bob might believe their cryptosystem is secure because its details are unknown, when actually it can be easily compromised when Eve learns the encryption algorithm.

Instead, it is typically assumed that an adversary knows the encryption algorithm being used in a cryptosystem. As far back as 1883, Kerckhoffs [38] suggested that a cryptosystem should be secure even if everything, except the key, is known by the adversary (which has become referred to as Kerckhoffs' principle, reiterated later by Shannon [77]). The security of a cryptosystem then depends on the secrecy of the encryption key.

Cryptanalysis is the study of cryptosystems by attempting to break them, which depends on the information available to the cryptanalyst. In a "ciphertext-only" attack, the cryptanalyst sees only the encryption algorithm and ciphertext. In a known plaintext attack, the cryptanalyst can see the plaintext and corresponding ciphertext. A more powerful attack can be carried out if the cryptanalyst has some control over the plaintext or ciphertext. In a "chosen plaintext" attack, the cryptanalyst can choose the plaintext and see the corresponding ciphertext. In a "chosen ciphertext" attack, the cryptanalyst can choose the ciphertext and see the corresponding plaintext.

It is current practice to publish encryption algorithms for public and expert review. Experimental evaluation can identify weaknesses that might allow an adversary to crack an encrypted message faster than random guessing the key (i.e., a brute force attack). In theory, public review leads to the strongest algorithms being standardized. A strong cryptosystem may still be attacked by brute force guessing, but this can be thwarted simply by sufficiently long keys.

In practice, encryption algorithms do not have to be mathematically impossible to break, only impractical to break without an enormous amount of time or resources (another principle from Auguste Kerckhoffs). For instance, if a cryptosystem takes a thousand years to break, it is effectively secure because the plaintext will no longer be interesting after that time. Modern ciphers, as well as the secure protocols dependent on them, often make use of mathematical problems that are believed to be computationally difficult. If these problems turn out to be much easier to compute, then some secure protocols will be much less secure than believed.

Ciphers have been implemented in a variety of cryptographic devices throughout the long history of cryptography. Many cryptosystems were developed for military or political purposes, such as the famous rotor machines in World War II. Today, modern cryptography is widely available to consumers through hardware (specialized chips) and software (built into operating systems and applications). Secure end-to-end encrypted communications are widely used in wireless (cellular mobile networks and IEEE 802.11 wireless local area networks) as well as wired networks, e.g., Secure Socket Layer/Transport Layer Security (SSL/TLS) in the Internet.

This chapter is a concise introduction to the field of cryptography, and a review of its usage in WSNs, without requiring a deep understanding of mathematics. There are many good references on cryptography for the reader interested in more details, such as [10, 21, 27, 41, 55, 83]. Section 2 reviews symmetric or private key cryptography that was the only approach until the 1970s. Section 3 describes asymmetric key cryptography and its usefulness for digital signatures and digital certificates. Common uses of cryptography for confidentiality, key distribution, and authentication are highlighted in Sect. 4. Lastly, Sect. 5 is a short survey of cryptography used in WSNs. The section begins with a review of the unique constraints on cryptography in WSNs and then highlights the current literature dealing with evaluating ciphers (particularly elliptic curve cryptography), proposing new approaches for key management, and implementing identity-based cryptography (IBC).

2 Symmetric Key Cryptography

A symmetric or private key cryptosystem is shown in Fig. 2. The same key K is used for encryption and decryption. The plaintext M is encrypted into ciphertext $C = E_K(M)$, and the ciphertext is decrypted by $M = D_K(C)$. The security of the system depends on the secrecy of the key K. There must be a separate method for Alice and Bob to agree securely on the secret key, either physically or electronically. The problem of key distribution or key agreement is a major challenge for symmetric key cryptography. After all, the purpose of the cryptosystem is to provide a secure communication channel for Alice and Bob. Without a secure channel, how can they share a secret key; or if they already have a secure channel to share a secret key, why is the cryptosystem needed?

Early ciphers used simple substitution (mapping letters of the plaintext alphabet to letters of the ciphertext alphabet) or transposition (rearranging the order of letters). However, substitution and transposition ciphers can be broken by frequency analysis. It is known that certain letters and words occur more frequently than others in most languages. For example, the letter "e" occurs most frequently in English. Although substitution and transposition may change the frequency distributions of letters and combinations of letters, some statistical structure remains in the ciphertext that can help a cryptanalyst.

Intuitively, encryption is effective if the ciphertext does not have any statistical structure that gives clues about the plaintext or encryption key. Claude Shannon suggested the concepts of "confusion" and "diffusion" [77]. Confusion refers to a complex relationship between the ciphertext and key which means that the ciphertext will not help an eavesdropper discover the key. As a counterexample, consider a polyalphabetic substitution cipher with an encryption key "ab." The key is repeated "abababab…" The "a" in the key means that letters in odd positions in the plaintext will be shifted by zero (i.e., unchanged). The "b" in the key means that letters in even positions in the plaintext will be shifted by one; e.g., "a" will be replaced by "b"; "b" replaced by "c". In this case, there is little confusion for the eavesdropper, who only has to look at the separate frequency distributions of odd position letters and even position letters. The frequency distribution of odd position letters will appear normal, suggesting that the plaintext letters were not changed. The frequency distribution of even position letters will also appear normal but shifted by one.

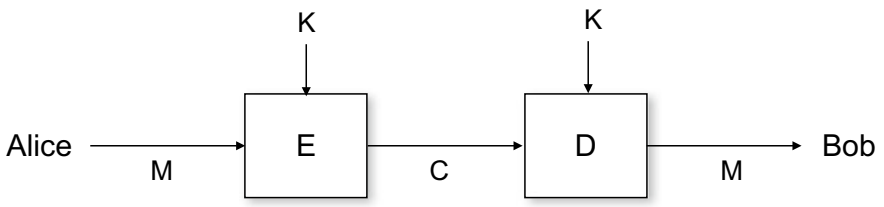


Fig. 2 Symmetric key cryptography

Thus, the eavesdropper may well guess that the key is "abababab…" In this case, the statistical structure of the ciphertext gives clues about the key.

Diffusion refers to the relationship between the plaintext and ciphertext. The opposite of diffusion is a one-to-one relationship between plaintext letters and ciphertext letters, e.g., a simple substitution cipher. The statistical structure of the plaintext will be reflected directly in the statistical structure of the ciphertext. Diffusion means that the statistical structure of the plaintext should be obscured by a complex dependency between plaintext letters and ciphertext letters. If a cipher has good diffusion, changing one plaintext letter will affect multiple letters in the ciphertext, or in other words, each ciphertext letter depends on many plaintext letters. A strict avalanche effect is a desirable property where a change of any plaintext letter will affect all ciphertext letters randomly and independently, changing any ciphertext letter with fifty percent probability.

Shannon further suggested that confusion and diffusion could be accomplished by a combination of substitutions and permutations, repeated multiple times. A product cipher can be constructed by multiple stages, each stage (or round) consisting of substitutions and permutations. This idea is somewhat surprising because it implies that a strong encryption algorithm does not necessarily require complex operations. Each stage may not achieve much confusion or diffusion by itself, due to the simplicity of substitutions and permutations. However, sufficient confusion and diffusion can be realized by a large number of stages. The US standards Data Encryption Standard (DES) and AES are based on this approach of constructing strong ciphers from multiple stages repeating simple operations.

2.1 Block Ciphers

Cryptosystems are categorized as block ciphers or stream ciphers. Block ciphers divide the plaintext message into fixed-length chunks and encrypt them separately, whereas stream ciphers encrypt one letter of plaintext at a time. The best known examples of block ciphers are DES, the US standard from 1977 to 2001, and its replacement AES. DES operates on 64-bit blocks using 56-bit keys (or 64 bits including 8 parity bits). In AES, the block size is 128 bits using three key sizes: 128, 192, or 256 bits, respectively.

2.1.1 DES

National Institute of Standards and Technology (NIST) adopted DES as the US standard based on the Lucifer cipher submitted by IBM led by Horst Feistel [61]. The general structure of DES encryption is shown in Fig. 3. DES consists of 16 rounds, each dependent on a 48-bit subkey (or round key) derived from the encryption key according to a key schedule. Thus, the "hardware" for each round is identical, but each round carries out different operations due to the different subkeys.

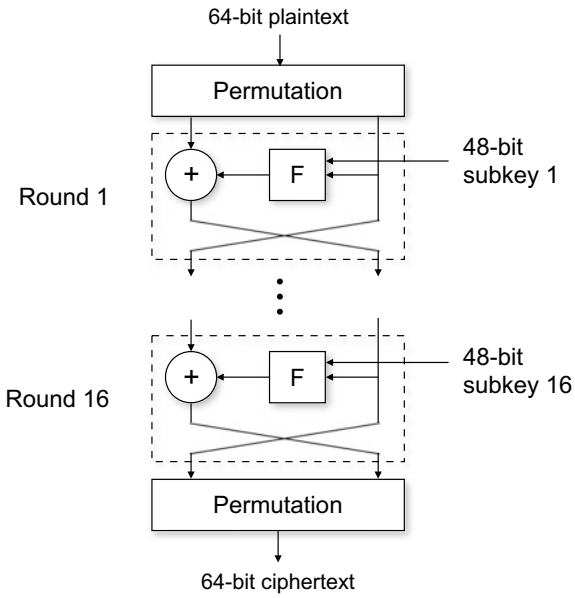


Fig. 3 General structure of DES

DES includes an initial and final permutation, but they are not important to the cryptographic function. The 64-bit plaintext block is divided into left and right halves, and then each round consists of these operations:

1. The right half goes through a *F* function.
2. The result is added (i.e., XOR'ed) with the left half.
3. The halves are transposed.

The *F* function is shown in Fig. 4. It carries out these operations:

1. The 32-bit left half text is expanded to 48 bits (by replicating some bits).
2. The result is added to the 48-bit subkey for that round.
3. The 48-bit result is changed to 32 bits by S-boxes that perform substitutions by lookup tables.
4. The 32-bit result goes through a permutation (P-box).

The S-boxes carry out a nonlinear transformation on 6-bit inputs and produce 4 bits of output. Their design is important to the strength of DES. It was rumored that the S-boxes in the original IBM design might have been changed by the National Security Agency (NSA) to implement a backdoor that could allow the NSA to break DES-encrypted messages easily. However, no backdoor has not been found. Instead, it has been found that the S-boxes are surprisingly resistant to an attack called differential cryptanalysis, which was proposed by Biham and Shamir in 1990 [8]. In 1994, IBM revealed they had been aware of differential cryptanalysis in the 1970s and subsequently changed the design of the S-boxes, which was kept confidential by request of the NSA [20].

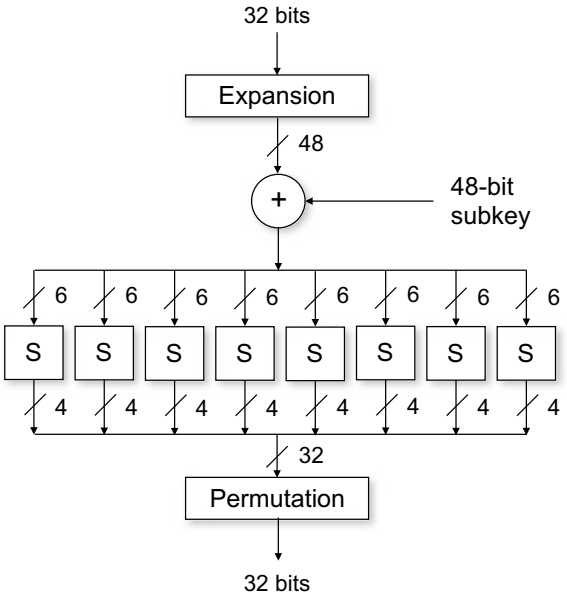


Fig. 4 F function in DES

As shown in Fig. 5, the key schedule consists mainly of shifting the bits of the encryption key and choosing a 48-bit subkey in each round. The 56-bit encryption key is first divided into two 28-bit halves. In each round, the halves are circularly shifted by 1 or 2 bits, and 48 bits are selected for the subkey. For details of the key schedule and S-boxes, refer to the DES standard specification [61].

DES has important practical advantages. First, the rounds can be implemented in identical hardware, simplifying implementation. Second, the substitutions and permutations are simple operations and fast in hardware. Third, the same hardware is used for encryption and decryption (a feature of Feistel ciphers); decryption is the same process as encryption except the order of subkeys is reversed.

Although no serious weakness was found in DES, it became apparent that the 56-bit key length was too short against brute force attacks as computing power was increasing rapidly in the 1990s. For instance, the DES Challenge III contest in 1999 to break a DES ciphertext was won by distributed.net (a distributed computing project) and a specialized computer called DES cracker ("Deep Crack") in just 22.5 h.

2.1.2 Triple DES

Double DES consisting of two stages of DES encryption using two separate keys would seem to offer the strength of 112-bit encryption. However, double DES was found to be vulnerable to a "meet-in-the-middle" attack [51]. As a result, double DES is not considered to be much stronger than DES.

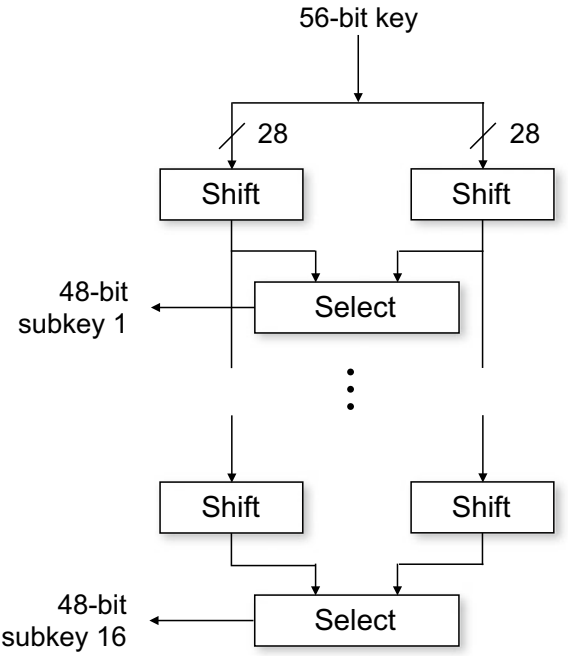


Fig. 5 DES key schedule

Triple DES consisting of three stages of DES encryption is not vulnerable to a meet-in-the-middle attack but offers only 112-bit security. Triple DES was recommended for a short time as a temporary method to achieve strong encryption resistant to brute force attacks until a replacement for DES could be found.

2.1.3 AES

NIST announced a contest to find a new US standard in 1997. The Rijndael cipher by Vincent Rijmen and Joan Daemen was chosen in October 2000 and standardized as AES in 2001 [62, 72]. The three versions of AES are listed in Table 1.

Like DES, AES consists of a number of identical rounds, each dependent on a different subkey (except the last round does not include the *MixColumns()* transformation). It is not a Feistel cipher like DES but a substitution–permutation network. The input and output of each round is an intermediate state represented by

Table 1 Versions of AES

Version	Key size (bits)	Rounds
AES-128	128	10
AES-192	192	12
AES-256	256	14

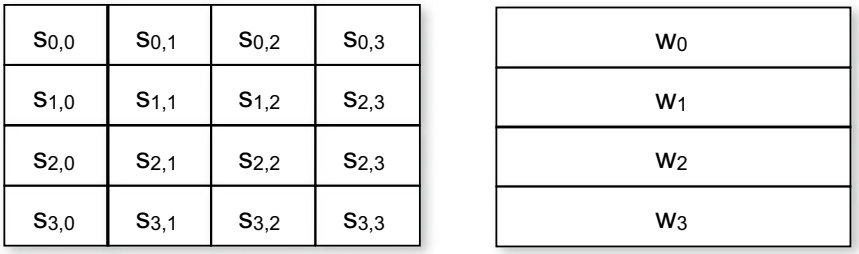


Fig. 6 AES state as array of bytes or column of words

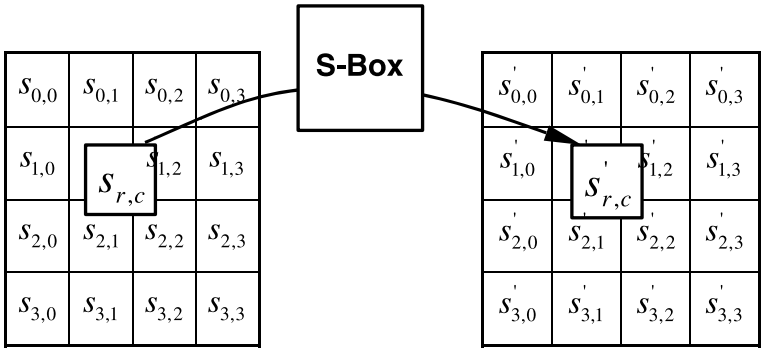


Fig. 7 SubBytes() transformation (from [62])

a 4 by 4 array of bytes $\{s_{0,0}, \ldots, s_{3,3}\}$, or a column of 4-byte words $\{w_0, \ldots, w_3\}$, as shown in Fig. 6. It is initialized as the 128-bit plaintext before the first round and changes after each round.

Each round carries out four transformations—*SubBytes()*, *ShiftRows()*, *MixColumns()* (except in the final round), and *AddRoundKey()*—described below.

1. *SubBytes()*: S-boxes perform substitutions of each byte of the state according to lookup tables, as shown in Fig. 7, which is a nonlinear transformation.
2. *ShiftRows()*: Transposition is performed by cyclically shifting each row as shown in Fig. 8. The first row is not shifted. The second row is shifted to the left by one byte. The third row is shifted to the left by two bytes, and the fourth row is shifted to the left by three bytes.
3. *MixColumns()*: Each column is multiplied by a fixed polynomial, shown in Fig. 9, effectively carrying out a matrix multiplication.
4. *AddRoundKey()*: Each column of the state is added (XOR'ed) to a 4-byte word of the subkey (for that round), as shown in Fig. 10.

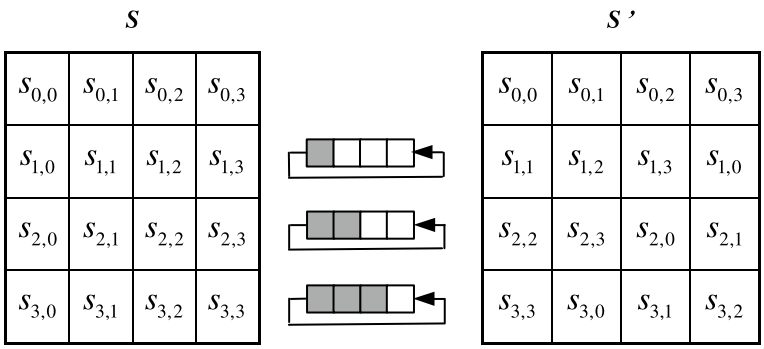


Fig. 8 ShiftRows() transformation (from [62])

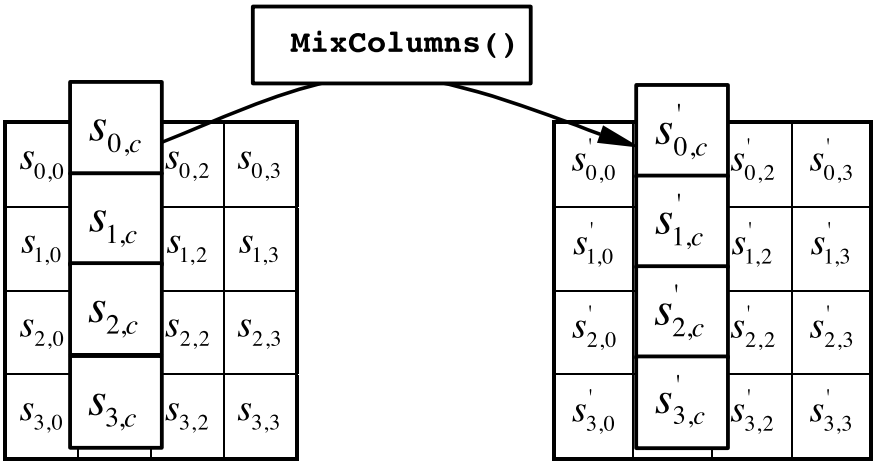


Fig. 9 MixColumns() transformation (from [62])

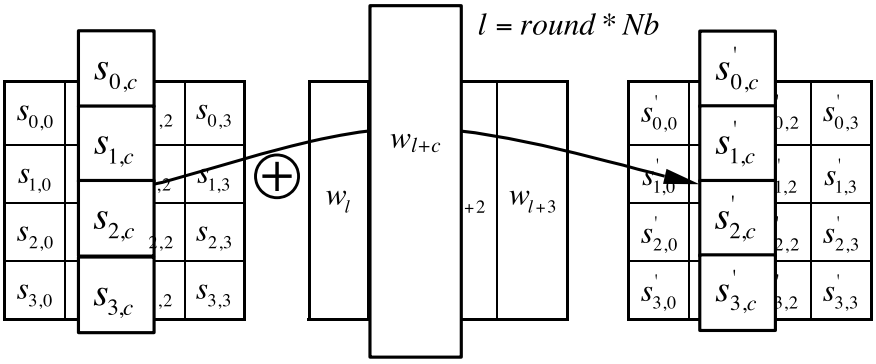


Fig. 10 AddRoundKey() transformation (from [62])

The S-boxes are designed mathematically based on the mathematics of Galois fields to provide good confusion and diffusion. First, S-boxes should exhibit a strict avalanche property: Changing one input bit will change or not change each output bit with equal likelihood. The bit independence criterion states that output bits should also change independently. Second, S-boxes should have balance; i.e., the numbers of 0 and 1 bits in the output should be equally frequent (for any arbitrary input).

The AES key schedule generates 128-bit subkeys for each round from the 128-bit key. First, there is a complicated key expansion using a combination of rotation, exponentiation, and substitutions. It is followed by choosing a subset from the expanded key. For more details, refer to the AES standard specification [62] or Rijmen and Daemen's description of Rijndael [72].

2.1.4 Block Cipher Modes

The most obvious way to use a block cipher is to encode each plaintext block separately, called the electronic codebook (ECB) mode. A cipher is essentially a mapping of a given plaintext to a specific ciphertext. It is called the codebook mode because it is theoretically possible to list all plaintext–ciphertext pairs in a table (codebook). A drawback to the ECB mode is that two identical plaintext blocks will produce identical ciphertext blocks, which can be helpful to cryptanalysis.

Another popular mode is cipher block chaining (CBC) shown in Fig. 11. In this mode, each plaintext block is XOR'ed with the previous ciphertext block before being encrypted. This prevents two identical plaintext blocks producing identical ciphertext blocks; clearly, each ciphertext block depends on all previous blocks.

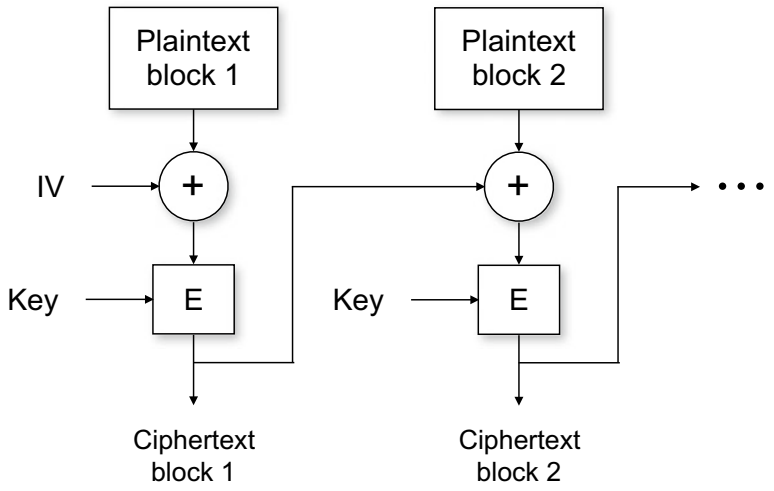


Fig. 11 CBC mode

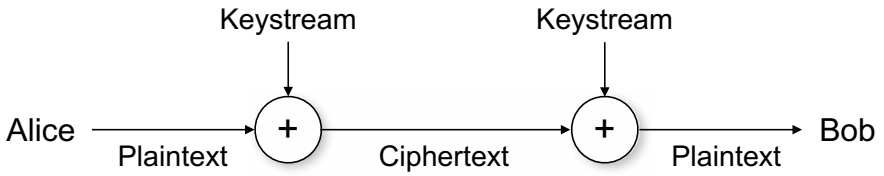


Fig. 12 A stream cipher

The first block has no previous ciphertext block; instead, the plaintext block is XOR'ed with a chosen initialization vector (IV). The drawback to the CBC mode is that encryption must be done serially. It is not possible to encrypt all blocks in parallel, which can be done ECB mode.

2.2 Stream Ciphers

Stream ciphers encode one letter of plaintext at a time. The basic operation is shown in Fig. 12. For encryption, a pseudorandom keystream is XOR'ed bit-by-bit with the plaintext. The same pseudorandom keystream is XOR'ed with the ciphertext at the decryptor. Stream ciphers are simple to implement and work with any plaintext size (unlike block ciphers).

The concept may be traced to Gilbert Vernam's 1919 invention (US patent 1,310,719) where the keystream was encoded in teletype tape. The keystream was bitwise XOR'ed with the plaintext at the encryptor. The decryptor carried out the same operation using the same tape.

2.2.1 One-Time Pad

Joseph Mauborgne in the US Army Signal Corps noted that if the keystream was completely random and never reused, then cryptanalysis would be very difficult. If the keystream is truly random, the ciphertext will be completely random and impossible to break. Suppose the keystream bit is 0 or 1 with equal likelihood; then, the ciphertext bit will be 0 or 1 with equal likelihood for any plaintext. This is an embodiment of a "one-time pad" which is considered to be totally secure.

A one-time pad has three requirements: (1) The key is as long as the plaintext, (2) the key is completely random, and (3) a key is used only once. Claude Shannon referred to the one-time pad as "perfectly secure" in the sense that an adversary observing an infinite amount of ciphertext is still left uncertain about the corresponding plaintext [77].

Consider a case where the plaintext is n bits and the key is also n bits. A given key K specifies one of 2^n mappings of a plaintext M to a ciphertext C. If all 2^n possible

keys are equally likely, then a given M will be mapped with equal likelihood to any of 2^n possible C. From an adversary's a priori view, all 2^n possible M are equally likely (before seeing any ciphertext). After an adversary observes a particular ciphertext C, the adversary's a posteriori view is the same as the a priori view: All 2^n possible M are still equally likely to have produced the observed C. Hence, observations of ciphertext do not give any useful information to cryptanalysis.

2.2.2 Linear Feedback Shift Registers

A stream cipher with a truly random keystream is not practical because the encryptor would need to send the random keystream (which is as long as the plaintext) securely to the decryptor, requiring a separate secure channel. Instead, stream ciphers use a pseudorandom bitstream which is a deterministic sequence that appears to be random in the sense that it should satisfy many statistical tests for randomness (e.g., 0 and 1 bits are equally frequent; autocorrelation is zero for all lags). The encryptor needs to only synchronize the initial state of the pseudorandom number generator with the decryptor.

Pseudorandom bitstreams are often generated by linear feedback shift registers (LFSRs). An example is shown in Fig. 13. The shift register holds three bits, and the feedback to the input (at s_2) is the sum of bits s_2 and s_0. Suppose the shift register holds 111. The bits will be shifted to the right by one, and the feedback will be a 0 bit input at the left. The shift register will then hold 011 at the next clock tick. As the clock ticks, the shift register contents will change to 101, then 010, 001, 100, 110, and back to 111. This LFSR has a period of 7 (the contents 000 is not allowed because it will continue to be 000 indefinitely). LFSRs with n bits have a maximum period of $2^n - 1$, and LFSRs with the maximum period are practically interesting because they produce the longest pseudorandom sequences (before the numbers start to repeat).

The simple structures of LFSRs make them vulnerable to attacks that can guess their structure from observing enough output. Nonlinear feedback shift registers are more resistant to attacks. It is common practice to construct them by combining multiple LFSRs with a nonlinear function. For example, this approach is taken for the stream ciphers A5/1 and A5/2 used in Global System for Mobile Communications (GSM), a standard for second-generation mobile cellular networks.

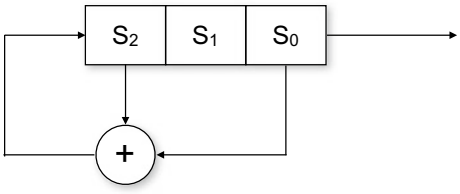


Fig. 13 Example of a 3-bit LFSR

2.2.3 RC4

RC4 by Ron Rivest is the best known and widely used stream cipher (e.g., in IEEE 802.11 WEP and BitTorrent). It makes use of a state register consisting of 256 bytes denoted $S[0], \ldots, S[255]$. The state register is initialized by the following procedure. First, the state begins as $\{0, 1, \ldots, 255\}$. A key register denoted $K[0], \ldots, K[255]$ is initialized with a key (if the key is shorter than 256 bytes, it is repeated up to 256 bytes). Then, the state register is permuted depending on the key register [27]:

```
j=0
for i = 0 to 255
{
    j = (j + S[i] + K[i]) mod 256
    swap (S[i], S[j])
}
```

After initialization of the state register, the keystream K can be generated by the pseudocode repeated for each bit [27]:

```
i = (i + 1) mod 256
j = (j + S[i]) mod 256
swap (S[i], S[j])
t = (S[i] + S[j]) mod 256
K = S[t]
```

2.3 Key Exchange

The main drawback of symmetric key cryptography is the need for Alice and Bob to agree somehow on a secret encryption key through physical or electronic means. Obviously, it would be more convenient to be able to share secret keys through a network instead of sharing them physically, e.g., in person or through mail. However, a key agreement protocol introduces the risk that the secret key might be exposed to adversaries through eavesdropping or masquerade attacks. Indeed, since standardized encryption algorithms such as DES and AES are believed to be strong, adversaries will look intently for weaknesses in the way keys are exchanged or stored.

2.3.1 KDC

Alice and Bob may share secret keys with or without the help of a trusted third party. A key distribution center (KDC) is a trusted third party assumed to have a preexisting trust relationship with Alice and Bob. The KDC is assumed to share

separate secret keys with Alice and Bob so that they can communicate securely with the KDC. Protocols such as Needham–Schroeder [60] and Kerberos [1] allow a KDC to securely distribute a session key for Alice and Bob to communicate securely with each other.

However, a KDC is a centralized approach and not scalable to large networks. For instance, Kerberos is not designed to work over a wide area network. The KDC represents a performance bottleneck because it is involved in every new session. It is also a single point of failure or compromise.

2.3.2 Diffie–Hellman

Diffie–Hellman is a well-known key agreement protocol that allows Alice and Bob to agree on a secret key directly without a third party such as a KDC [24]. Without prior agreement on a secret, Alice and Bob exchange public numbers and calculate the same private key. An eavesdropper Eve observes the same public numbers but is unable to discover the private key. The Diffie–Hellman protocol is based on the mathematical difficulty of finding discrete logarithms of large numbers.

The discrete logarithm is an example of a one-way function that is easy to calculate but the inverse function is extremely difficult. Given x, g, and p, the calculation of

$$y = g^x \bmod p \tag{1}$$

is fairly easy. That is, y is the remainder after dividing g^x by p. However, given y, no efficient method is known for finding x if g and p are chosen carefully, namely p is a prime number and g is a primitive root mod p. Then x must be found by guessing, which becomes computationally infeasible if x is a large number (at least 100 digits) and p is a large prime (at least 300 digits).

Assume that p is a (large) prime number and g is an integer ($g < p$) and a primitive root mod p. Both p and g may be known publicly. The Diffie–Hellman key agreement protocol proceeds with Alice and Bob each choosing secret numbers and exchanging public numbers:

- Alice chooses an arbitrary secret A ($A < p - 2$) and sends $A' = g^A \bmod p$ to Bob.
- Bob chooses an arbitrary secret B ($B < p - 2$) and sends $B' = g^B \bmod p$ to Alice.
- Alice computes $k = (B')^A \bmod p$.
- Bob computes $k' = (A')^B \bmod p$.

Alice and Bob's shared secret key is $k = k' = g^{AB} \bmod p$. They find the same key because Alice computes

$$(B')^A \bmod p = (g^B)^A = g^{AB} \bmod p \tag{2}$$

and Bob computes

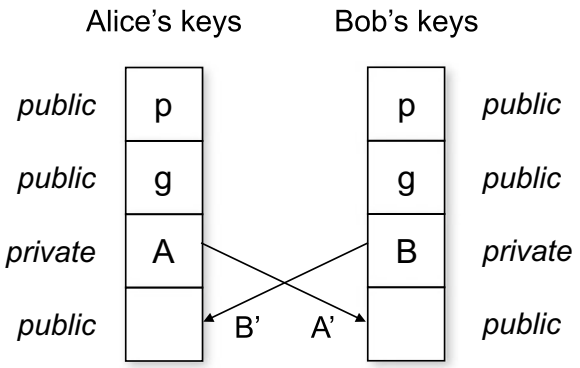


Fig. 14 Diffie–Hellman protocol from the perspective of public and private keys

$$(A')^B \bmod p = (g^A)^B = g^{AB} \bmod p \tag{3}$$

The protocol is shown in Fig. 14 from the viewpoint of public and private keys. Alice has public keys p, g, and B' (from Bob) and private key A. Likewise, Bob has public keys p, g, and A' (from Alice) and private key B. Although their keys are different, they compute the same secret k.

Consider an example with $p = 353$ and $g = 3$.

- Suppose Alice chooses a secret $A = 97$ and sends $A' = 3^{97} \bmod 353 = 40$.
- Bob chooses a secret $B = 233$ and sends $B' = 3^{233} \bmod 353 = 248$.
- Alice computes $k = 248^{97} \bmod 353 = 160$.
- Bob computes $k' = 40^{233} \bmod 353 = 160$.

Their shared secret key is $k = k' = 160$.

In this example, an eavesdropper Eve will know $p = 353$, $g = 3$, $A' = 40$, and $B' = 248$. From these numbers, Eve wants to find A or, equivalently, B. Eve knows that $40 = 3^A \bmod 353$; she can try different guesses for A, but no easy way to find A is known.

3 Asymmetric Key Cryptography

The concept of asymmetric or public key cryptography was speculated for many years, but no practical public key cryptosystem was known until the invention of Rivest–Shamir–Adleman (RSA) in 1977 [74]. As mentioned earlier, the main drawback of symmetric key cryptography is the need for Alice and Bob to share a secret key, risking exposure of the key to Eve. Asymmetric or public key cryptography avoids the need for a shared secret.

An asymmetric key cryptosystem is shown in Fig. 15. Bob has a public key K_{pub} to be shared with anyone and a private key K_{priv} that is never shared. Although the two keys are mathematically related, it should be computationally infeasible to

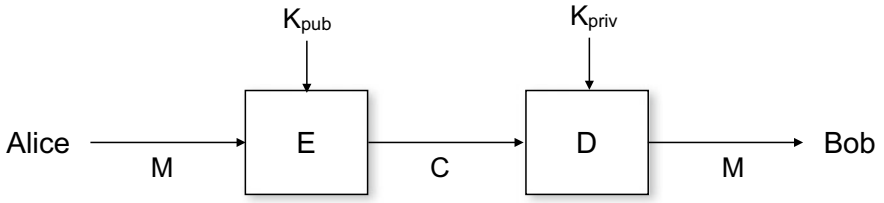


Fig. 15 Asymmetric key cryptography

discover the private key from the public key. A message for Bob is encrypted using his public key as $C = E_{K_{pub}}(M)$, and Bob decrypts the ciphertext using his private key as $M = D_{K_{priv}}(C)$.

3.1 RSA

RSA depends on the longstanding belief in mathematics that it is easy to multiply two prime numbers, but difficult to factor the product into two primes particularly when the prime factors are very large. It makes use of modular arithmetic which follows notation and rules that may seem unusual compared to everyday mathematics. For example:

- Let X be a number in modular arithmetic, then it is an integer between 0 and $n - 1$ (if mod n).
- X mod n is the remainder of X divided by n. For example, $12 \bmod 10 = 2$ or $12 = 2 \bmod 10$.
- $-X$ is the additive inverse where $X + (-X) = 0 \bmod n$. For example, if $X = 4$ then $-X = 6 \bmod 10$ because $4 + 6 = 0 \bmod 10$.
- X^{-1} is the multiplicative inverse where $X(X^{-1}) = 1 \bmod n$. For example, if $X = 7$, then $X^{-1} = 3 \bmod 10$ because $7 \times 3 = 1 \bmod 10$.
- The totient function $\phi(n)$ represents how many numbers less than n that are relatively prime to n. If $n = pq$ where p and q are prime numbers, then $\phi(n) = (p - 1)(q - 1)$.

RSA keys are generated by these steps:

1. Choose two secret prime numbers p and q (which will be discarded after the keys are created).
2. Calculate the modulus $n = pq$.
3. Calculate the totient $\phi(n) = (p - 1)(q - 1)$.
4. Select an integer e less than $\phi(n)$ that is relatively prime to $\phi(n)$.
5. Find d as the multiplicative inverse of e mod $\phi(n)$, i.e., such that $de \bmod \phi(n) = 1$.

The public key is $\{e, n\}$, and the private key is $\{d, n\}$.

RSA can use any key length and any plaintext size (shorter than the key). The ciphertext will determine the key length. RSA encryption and decryption are actually simple calculations. To encrypt a message $M < n$, the ciphertext is computed as

$$C = M^e \bmod n. \tag{4}$$

Given ciphertext C, it is decrypted by computing

$$M = C^d \bmod n. \tag{5}$$

Consider an example with $p = 61$ and $q = 53$. The modulus is $n = 61 \times 53 = 3233$, and totient is $\phi(3233) = 60 \times 52 = 3120$. Suppose the public exponent chosen is $e = 17$. Confirm that the private exponent $d = 2753$ is the multiplicative inverse of e by calculating $de \bmod \phi(3233) = 17 \times 2753 \bmod 3120 = 46801 \bmod 3120 = 1$. The public key is $\{17, 3233\}$, and the private key is $\{2753, 3233\}$.

Continuing this example, the encryption and decryption process can be illustrated with a message, say $M = 123$. The ciphertext is:

$$C = M^e \bmod n = 123^{17} \bmod 3233 = 855. \tag{6}$$

The message is recovered by decrypting:

$$M = C^d \bmod n = 855^{2753} \bmod 3233 = 123. \tag{7}$$

Why does RSA encryption and decryption work? Start with the ciphertext $C = M^e \bmod n$, the following shows that the message can be recovered by decryption.

$$\begin{aligned} M &= C^d \bmod n && (8)\\ &= (M^e \bmod n)^d \bmod n && (9)\\ &= (M^e)^d \bmod n = M^{ed} \bmod n && (10)\\ &= M^{1 \bmod \phi(n)} \bmod n && (11)\\ &= M \bmod n = M \text{ if } M < n && (12) \end{aligned}$$

If p and q are known, then decryption is easy because the totient $\phi(n)$ can be calculated, which then leads to finding the private exponent d. However, p and q are discarded after the RSA keys are created. An adversary does not know p and q but can find them by factoring n. However, as mentioned earlier, factoring the product of large prime numbers is a known difficult problem. In practice, p and q should each be at least 150 decimals.

3.2 Hash Functions

Asymmetric key cryptography makes it possible to digitally sign messages to verify the sender's identity and provide assurance that the message has not been altered. Digital signatures depend on cryptographic hash functions; the design of good cryptographic hash functions is a specialized field of study [50].

Hashes can be motivated by the well-known use of checksums appended to messages for error detection. Checksums are commonly parity bits or cyclic redundancy check (CRC) codes [69]. These are designed to detect or correct random bit errors that might occur during message transmission. However, they are not well suited to protect against an adversary who might deliberately alter a message in transit and recompute a new checksum. Instead of a checksum, better protection would be a message authentication code (MAC) that acts like a checksum or "message digest" but dependent on a secret key [36]. An adversary could not alter a message and attach a new MAC without knowing the secret key. Hash functions such as MD5 or SHA-1 have been developed with certain properties to be well suited for MACs.

Some desirable properties for hash functions include:

- Message M can be any length, whereas its hash $H(M)$ is a fixed length.
- $H(M)$ is easy to calculate for any message M.
- Pre-image resistance: Given a hash h, it is computationally infeasible to find any message M such that $H(M) = h$. Like ciphertext, hashes generally look like random bit strings with no apparent statistical structure giving clues about the original message.
- Second pre-image resistance: Given M and $H(M)$, it is computationally infeasible to find any M' such that $H(M') = H(M)$.
- Collision resistance: It is computationally infeasible to find different M and M' such that $H(M) = H(M')$.

Generally, messages are longer than the hash, so it is not possible to recover the original message. Suppose messages are n bits, and hashes are m bits ($m < n$). Thus, the hash function maps 2^n messages to a smaller number of 2^m hashes. On average, 2^{n-m} messages are mapped to the same hash. Hence, some information is lost in a sense when a message is "boiled down" to a hash. It is not possible to reverse the hash function because a given hash might feasibly have been produced by many possible messages.

Intuitively, a hash might be thought of as a "fingerprint" of a message; however, a hash is not quite a unique message identifier like a real fingerprint. Again, if messages are n bits, and hashes are m bits ($m < n$), then 2^{n-m} messages produce the same hash on average.

It is clear that resistance properties are highly dependent on the length of the hash. How much effort is involved in finding a message that will match a given hash? Each guess has a probability of 2^{-m} of matching the hash. The probability of at least one match in j guesses is $1 - (1 - 2^{-m})^j$; if this probability is 0.5, then the required number of guesses is $j = 2^{m-1}$ when m is large.

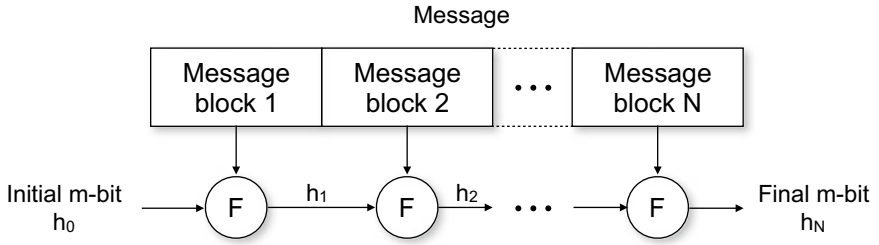


Fig. 16 Construction of hash functions by repeating compression function F

One of the applications of hashes is verification of passwords. Instead of storing passwords on a computer system, which might be stolen, some computer systems store the hashes of passwords. When a user submits a password, it is hashed and compared to the stored hash for a match. If an adversary can keep making guesses, the adversary will eventually find a guess to match the stored hash. Most computer systems have a limit of three failed login attempts, in which case, the probability of success for the adversary is $1 - (1 - 2^{-m})^3$.

Another application of hashes is verification of a file's data integrity. Suppose the hash of a file is stored separately. If the file is accessed and changed illegitimately, it will not match its stored hash. However, it might have been possible that an adversary changed the message to a different message that matches the hash. In this case, the second pre-image resistance property is important. The adversary is able to see the original file M and compute its hash h, and wants to find another file M' that has the same hash. It is similar to the earlier effort; the probability of at least one match in j guesses is $1 - (1 - 2^{-m})^j$.

Damgard and Merkle proposed a general method to construct good hash functions from repeated stages of a "compression function" F as shown in Fig. 16 [50]. The compression function takes a fixed-length input and produces a shorter fixed-length output. Any size message can be handled by repeating more stages. The message is divided into blocks. In the first stage, the first message block and an initial set of bits are put into the first compression function. The result and second message block are input into the second compression function, and so on. The compression function can be a block cipher or a special purpose function as in MD5 and SHA-1.

In Message Digest 5 (MD5) by Ron Rivest, any length message is divided into 512-bit blocks. The compression function takes 128 + 512 bits input and produces a 128-bit output. The final result is a 128-bit hash. For details of the MD5 compression function, refer to IETF RFC 1321 [73]. Unfortunately, serious flaws have been found in MD5 and its usage is generally discouraged.

NIST standards include a family of Secure Hash Algorithm (SHA) functions consisting of SHA-1, SHA-2, and SHA-3. Their differences are listed in Table 2. Standardized in 1995, SHA-1 is a 160-bit hash function resembling MD5 [63]. The 2001 SHA-2 standard includes SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256 [63]. Standardized in 2005, SHA-3 is a hash function

Table 2 SHA family of hash functions

Hash	Output (bits)	Max. message length (bits)	Block size (bits)	Rounds
SHA-1	160	$2^{64} - 1$	512	80
SHA-224	224	$2^{64} - 1$	512	64
SHA-256	256	$2^{64} - 1$	512	64
SHA-384	384	$2^{128} - 1$	1024	80
SHA-512	512	$2^{128} - 1$	1024	80
SHA-512/224	224	$2^{128} - 1$	1024	80
SHA-512/256	256	$2^{128} - 1$	1024	80
SHA3-224	224	None	1152	24
SHA3-256	256	None	1088	24
SHA3-384	384	None	832	24
SHA3-512	512	None	576	24

formerly called Keccak, with the same hash lengths as SHA-2 but with a different internal structure [64]. For details, refer to the NIST standards.

3.3 HMAC

A hash $H(M)$ appended to a message M does not provide more security than a checksum. An adversary can change the message and append a new hash. A solution could be to encrypt the hash using a secret key K, but an even simpler and more efficient approach is to compute a hash of the message concatenated with the secret key, i.e., $H(M, K)$. An adversary cannot change the message and compute a new hash without knowing the secret key.

An example of this approach is the MAC used in Secure Socket Layer/Transport Layer Security (SSL/TLS) standardized by the Internet Engineering Task Force (IETF) [23]. SSL adds end-to-end security to the transport layer protocol transmission control protocol (TCP) for secure Web connections. During the SSL handshake protocol between a Web server and client, they negotiate an agreement on a pair of message authentication keys. When data is sent, it is segmented into fragments, and each fragment is appended with a MAC. The MAC is essentially a MD5 or SHA-1 hash of a concatenation of the fragment, message authentication key, sequence number, and fragment length. The MAC verifies the sender's identity and data integrity of the fragment.

Hash-based message authentication code (HMAC) is an IETF recommendation [42] for computing a MAC dependent on a secret key using any hash function as an input. As shown in Fig. 17, an HMAC is computed and appended to a message before transmission. It requires a secret key to be pre-agreed between Alice and Bob. The

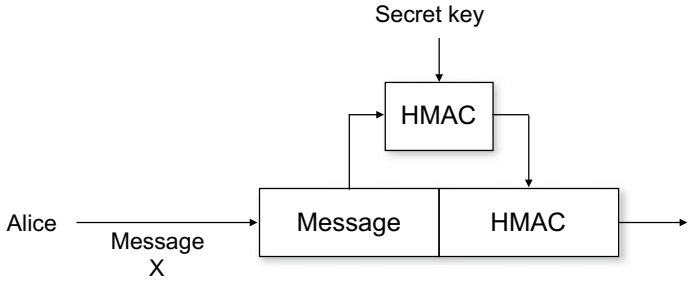


Fig. 17 Adding HMAC to a message

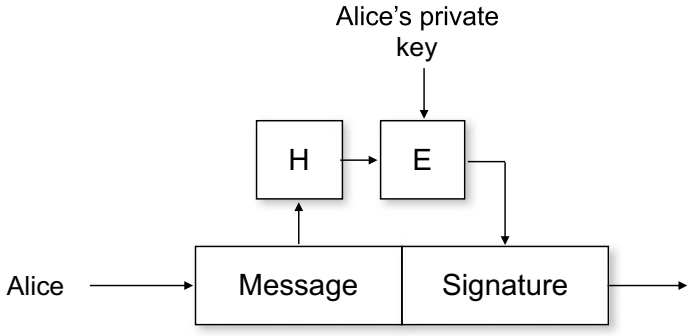


Fig. 18 Adding a digital signature to a message

calculation of HMAC is a complicated algorithm involving two stages of hashing; details are in IETF RFC 2104.

3.4 Digital Signatures

HMAC is a straightforward method to provide sender authentication and data integrity assuming that Alice and Bob share a secret key. Digital signatures taking advantage of asymmetric key cryptography offer an alternative that avoids the need for a secret key [74]. Suppose Alice wants to digitally sign a message to Bob. She can compute a hash of the message, and then encrypt it using her private key to create a digital signature as shown in Fig. 18.

When Bob receives the signed message, he decrypts the digital signature using Alice's public key as shown in Fig. 19. The resulting hash is compared with a new hash that Bob computes from the received message. If they match, Bob is assured that the message has not been altered, and the message came from Alice (because the signature was decrypted properly using Alice's public key).

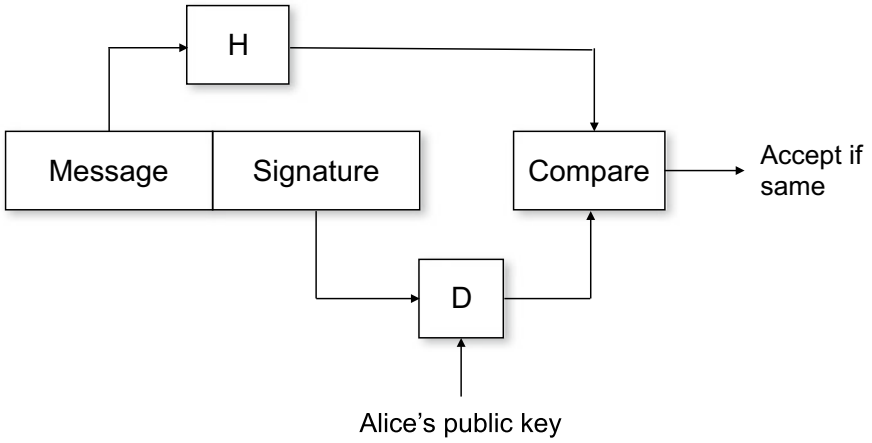


Fig. 19 Verifying a received message

Suppose that an eavesdropper Eve intercepts and changes Alice's message, or forges a different message pretending to come from Alice. Eve has to calculate a valid signature. She can compute a hash of the new or altered message but cannot encrypt the hash without Alice's private key.

A digital signature is appended to a message without changing the message. This means that a digital signature does not protect the message from eavesdropping, but the message can obviously be encrypted for confidentiality if wanted.

3.5 Digital Certificates

Digital signatures come in handy for public key certificates. At first glance, it may seem that asymmetric key cryptography solves the problem of key distribution because Alice and Bob do not need to share a secret key in order to communicate. Alice does not ever share her private key and can give her public key to everyone. However, it is not straightforward to securely distribute public keys with protection against tampering. Suppose Bob receives a public key that appears to belong to Alice; how can Bob verify it is really Alice's key?

Certificate authorities (CAs) are organizations acting as trusted third parties to vouch for ownership of public keys. Essentially, CAs issue public key certificates that declare the owner of a particular public key, along with the CA's digital signature. The owner can distribute certificates to anyone, and the CA's signature provides assurance that the certificate has not been forged or altered. For example, Web site operators present their public key in the form of a certificate during the SSL handshake protocol.

X.509 is an International Telecommunications Union (ITU) standard for public key certificates adopted by the IETF's Public Key Infrastructure X.509 (PKIX) working group [19]. The current version 3 certificate includes these fields:

- Version (3);
- Certificate serial number;
- Signature algorithm and parameters;
- CA's name;
- Dates of validity;
- Certificate owner's name;
- Owner's public key; and
- CA's digital signature.

4 Applications of Cryptography

People usually think of cryptography to protect data confidentiality. For example, when the padlock icon appears in a Web browser, the user is typically aware that means the session is encrypted to protect online banking or online shopping transactions from eavesdropping. But cryptography has broader applications including key distribution and user authentication. This is the reason that cryptography is the fundamental foundation for secure protocols.

4.1 Privacy

Encryption is particularly important in wireless networks where radio signals are easily vulnerable to eavesdropping. This section reviews some examples of encryption used in wireless networks.

IEEE 802.15.4: IEEE 802.15.4 is a technical standard for the lower protocol layers (physical and medium access control) for low-rate wireless personal area networks (LR-WPANs). The encryption algorithm used is AES-128.

ZigBee: IEEE 802.15.4 is the basis for ZigBee and several other wireless standards. ZigBee is a technical standard for low-power, low-bandwidth wireless personal area networks maintained by the ZigBee Alliance. ZigBee adds two security layers on top of 802.15.4 in the network and application layers. Building on IEEE 802.15.4, AES-128 is still the encryption algorithm. Three types of keys are defined: master, link, and network. Master keys are preinstalled in each node to protect the confidentiality of link key exchange. Link keys are exchanged between pairs of nodes for encrypting data in a session. The network key is a unique 128-bit number needed to joint the network. It is generated and regenerated by the "Trust Center" which may be the coordinator or a separate device.

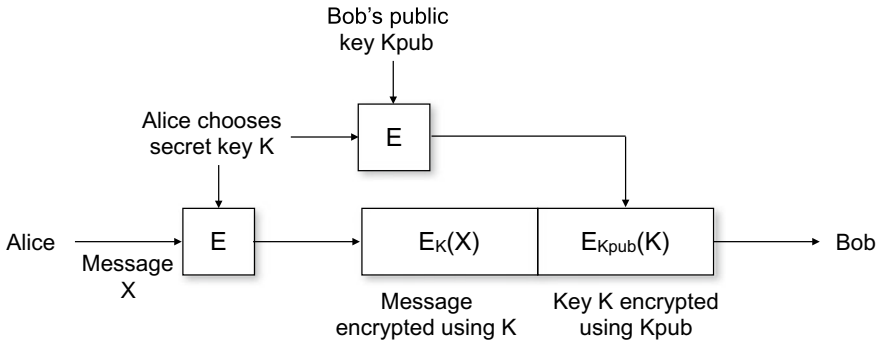


Fig. 20 Asymmetric key cryptography for key wrapping

IEEE 802.11 WPA/WPA2: IEEE 802.11i-2004 is an amendment added to the original IEEE 802.11 standard in 2004 that establishes WPA2 (Wi-Fi Protected Access) for Wi-Fi network security. It specifies the AES-128 block cipher for encryption. WPA was an interim standard prior to IEEE 802.11i that recognized the need for backward compatibility using RC4 but recommended AES as an option.

4.2 Key Wrapping by Asymmetric Key Cryptography

Asymmetric key cryptography offers an easy way to share secret keys. Symmetric key cryptography is preferred for long messages because it is orders of magnitude faster than asymmetric key cryptography such as RSA. As mentioned earlier, Alice and Bob can establish a secret session key by using Diffie–Hellman or a trusted third party (KDC). Instead, Alice can send an encrypted message along with a wrapped secret encryption key as shown in Fig. 20.

Alice chooses any secret encryption key K. The message is encrypted, say with AES, using key K. The key is also attached to the message encrypted with Bob's public key. Presumably, only Bob has his private key that can decrypt the wrapped secret key and then decrypt the message.

4.3 User Authentication

There are two authentication problems: verifying a message's sender and verification of a user's identity at the start of a new session. User authentication is performed only once per session, whereas message authentication is done for every message. User authentication can be done by symmetric or asymmetric key cryptography. In both cases, authentication protocols often use the concept of challenge–response.

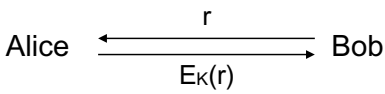


Fig. 21 Authentication by challenge–response

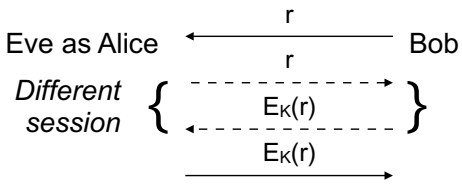


Fig. 22 Reflection attack

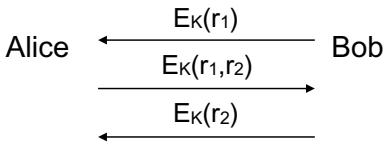


Fig. 23 Improved challenge–response

Suppose that Alice and Bob share a secret key K. Obviously, Alice and Bob can prove their identities by revealing K to each other, but then the secret would be exposed to an eavesdropper. The basic challenge–response protocol shown in Fig. 21 allows Alice and Bob to demonstrate their knowledge of K without exposing it to Eve. Bob challenges Alice with a random nonce r. Alice replies by encrypting the challenge using the secret key K, i.e., $E_K(r)$. Thus, Alice shows knowledge of K without revealing it. Bob's identity is authenticated similarly with a different challenge from Alice. Different variations are feasible; e.g., the response can be an HMAC of the challenge and the secret key.

It is well known that the basic challenge–response protocol may be vulnerable to a reflection attack. Suppose Eve tries to masquerade as Alice. As usual, Bob will send a random challenge r to Eve. Eve must reply with the encrypted challenge $E_K(r)$, but of course, she cannot reply without knowing the secret K. However, she can open another session with Bob and "reflect" the same challenge r to Bob as shown in Fig. 22. Bob will reply with the correct response, which Eve can then use as her response to Bob's earlier challenge. It might seem that Bob is dumb for responding to his own challenge but think of Bob as an automated software program, not a human being.

There are many possible ways to modify the challenge–response protocol to prevent a reflection attack. For instance, the response can include the responder's name. When Bob issues a challenge r to Alice, Alice can reply by encrypting ("*Alice*", r). If Eve is masquerading as Alice and reflects Bob's challenge r back to Bob, she cannot reuse his response which has his name in it.

Another variation is shown in Fig. 23. The first challenge to Alice is encrypted so that Eve cannot respond to it. Likewise, the response from Alice is encrypted and serves simultaneously as a response and challenge.

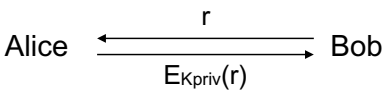


Fig. 24 Challenge–response using asymmetric key cryptography

Challenge–response works in a slightly different way with asymmetric key cryptography. Suppose Bob wants to verify Alice's identity. As usual, Bob issues a random challenge r. Alice can encrypt r using her private key as shown in Fig. 24. Bob can verify the response by decrypting it using Alice's public key and check that it correctly decrypts to r. This verifies that Alice possesses the private key used, which presumably only Alice should know.

Challenge–response with asymmetric key cryptography is amenable to implementation in a physical device such as an ID card or security token. The device stores the owner's private key in an unreadable memory space and responds to any challenge by encrypting it using the private key. On the downside, the device must be carried constantly, and a user risks losing access if the device is lost.

5 Cryptography in WSNs

WSNs translate the real world into the digital realm [22, 49]. They have a diverse range of military and civilian applications including environmental monitoring, security and surveillance, healthcare monitoring, industrial control processes, smart home monitoring, and battlefield monitoring. WSNs are designed specifically for their applications which often involve a remote, harsh, or hostile environment without human attendance [65]. WSNs are decentralized; nodes organize themselves and collaborate toward their common goal [87].

The use of cryptography in WSNs must consider their unique characteristics. WSN nodes are characterized by [43]:

- Low data rate sensing although nodes may vary in capabilities [22];
- Low-power embedded processors (typically 8 MHz microcontroller with less than 128 KB of instruction memory [75]);
- Limited random access and read-only memory (typically around 10 KB of RAM memory [75]);
- Low-rate, short-range wireless radio (10–100 Kbps, up to 100 m);
- Geopositioning systems;
- Battery power (often difficult to change or recharge); and
- Intelligent in-network processing, e.g., for data compression or signal processing.

5.1 Security Threats

The often hostile environment means that nodes are exposed to multiple threats such as [49]:

- Common attacks (e.g., eavesdropping, packet modification, replay);
- Denial of service attacks;
- Node capture or compromise;
- Impersonation attacks; and
- Protocol-specific attacks.

Several good references are available for comprehensive overviews of security problems and solutions in WSNs [13, 26, 46, 49, 65, 68]. Cryptography is not expected to solve every security problem, but it does provide a mathematical foundation for important primitives addressing the problems of privacy, authentication, and data integrity. For example, say sensor nodes make majority-based decisions as part of their collaborative information processing. A malicious node might inject packets to cause an entire cluster to generate inaccurate decisions [4]. Cryptographic authentication can help to solve this problem, although there is an issue that the malicious node might have legitimate credentials, so authentication is not a complete solution.

A considerable literature has grown around the issues of appropriate cryptographic techniques for WSNs. Surveys on the topic can be found in [44, 66, 75, 78, 82]. In this section, the literature is reviewed following these fundamental issues:

- Symmetric or asymmetric key encryption;
- Elliptic curve cryptography (ECC);
- Key management; and
- Identity-based cryptography (IBC).

5.2 Symmetric Versus Asymmetric Key Cryptography

Many studies have implemented symmetric and asymmetric key ciphers to evaluate their energy demands. It is well known that asymmetric key ciphers such as RSA are orders of magnitude more computationally demanding than a symmetric key cipher such as AES. Hence, symmetric key cryptography is generally considered to be well suited for resource-constrained WSN devices, whereas asymmetric key cryptography is often ruled out [49, 66].

RC5, RC4, and TEA were mathematically analyzed for overheads of frequently used basic operations [48]. The mathematical modeling was verified by simulations to be accurate. Some symmetric key encryption algorithms (e.g., RC4, RC5, Skipjack, IDEA, AES) have been evaluated experimentally on sensor nodes [17, 30, 44, 66].

RC4 and Skipjack were found to be the most effective algorithms, while other algorithms such as AES incur a slightly higher overhead. Skipjack, MISTY1, and

Rijndael were found to be storage and energy efficient [44]. Also, the output feedback mode for block ciphers was recommended for pairwise links but cipher block chaining (CBC) mode for group communications. Another study recommended SkipJack, RC4, and HIGHT for time-critical or energy-efficient applications, and TEA, SEA, and SkipJack if SRAM requirements are more important than time and energy [47].

To argue against the belief that asymmetric key cryptography is unsuitable for WSNs, RSA and Diffie–Hellman key agreement protocol were implemented in MICA2 motes running TinyOS [85]. It was concluded that asymmetric key cryptography with careful design can be deployed on even the most constrained sensor network devices. A comprehensive survey of asymmetric key cryptography for WSNs is found in [78].

An overview of both symmetric and asymmetric key cryptography for WSNs concluded that for asymmetric key cryptography, "very simple designs are able to efficiently provide ciphering and signing services for any kind of sensor node hardware" [75].

A broadcast scheme based on RSA was proposed [16]. Unlike traditional usage of RSA, the proposed scheme uses short moduli to enhance performance. Short moduli are normally avoided due to weak security, but the proposed scheme fixes that weakness by means of rekeying strategies. To minimize the rekeying overhead, a multi-modulus RSA generation algorithm is proposed.

5.3 ECC

The suitability of asymmetric key cryptography for WSNs has been uncertain because of costly key computation, long keys, and complex key distribution [31, 49, 66]. It has been noted that a public key is usually 1024 bits (128 bytes) or 2048 bits (256 bytes) long, making asymmetric key cryptography expensive in terms of computation (e.g., large integer modular exponentiations) and long packets [46].

However, a number of studies point out that elliptic curve cryptography (ECC) can be implemented much more efficiently than RSA [4, 5, 84]. ECC is theoretically appealing for offering high security for relatively small key sizes. It is often stated that the security of a 160-bit key for ECC is equivalent to 1,024-bit key of RSA. ECC is also attractive for memory and energy savings, and the simplicity of its underlying operation, the scalar point multiplication [31, 37, 49, 65, 84]. Point multiplication is a sequence of point additions and point doublings. Several techniques have been proposed to boost ECC point multiplication. For instance, an optimized dynamic window was shown to reduce average calculation time [35].

Koblitz [40] and Miller [53] independently introduced the notion of using elliptic curves in asymmetric key cryptography. ECC is based on algebraic structure of elliptic curves over finite Galois fields. These fields can be binary polynomial fields GF(2^n) or prime integer fields GF(P). ECC is a much harder problem than simply factoring integers. The security of ECC is based on the intractability of the elliptic curve version of the discrete logarithm problem.

Choosing binary fields is more attractive for hardware implementations because it reduces both hardware area and energy consumption. They also offer more options in terms of bases, irreducible polynomials, and fields.

An important issue with ECC is what are the best choices of the domain parameters of the elliptic curve for efficiency. An analysis has been done on the relevant parameters of elliptic curves for ECC implementation in WSNs and the impact of their values on the level of security they offer [59].

Theoretical expectations have been supported by a considerable amount of experimental results to show that ECC can be executed on resource-constrained sensor nodes in reasonable time. ECC has been implemented both in software [45, 86] and in hardware [7, 54, 58].

TinyECC is a widely used, configurable library for ECC operations including a number of optimization switches to change the state of the specific optimization as required by the need of the user [45]. The framework is flexible allowing different combinations of the optimizations resulting in different execution times and resource consumptions.

RSA and ECC operations (digital signatures and key exchange) were experimentally evaluated on four types of nodes: MICA2DOT, MICA2, MICAz, and TelosB [70]. They were found to have insignificant costs on lifetime. While both RSA and ECC are possible using 8-bit CPUs, ECC demonstrated a performance advantage over RSA [33].

An optimized ECC implementation was developed from scratch for an 8-bit ATmega128 microcontroller [18]. Measured performance was 2.78 times faster than the widely used TinyECC library.

A critical study of the underlying finite field, representation basis, occupied chip area, consumed power, and time performance was done in a hardware implementation of ECC [34].

The performance of bivariate polynomials was studied while changing the degree of the polynomial and the number of nodes [14].

Investigation of software implementations led to the recommendation for ECC defined on the standard binary Koblitz curve as the best choice for WSNs "in terms of computational complexity, communication overhead, key size, memory, and storage" [78].

ECC and RSA are not the only asymmetric algorithms. XTR-DSA and NTRUSign have been proposed as well as implemented for MICAz motes [25]. Pairing-based cryptography (PBC), related to ECC but more complex, was implemented on a number of platforms to show its feasibility [80].

ECC was compared with Rabins Scheme and NTRUEncrypt [31]. It was concluded that these asymmetric algorithms actually reduce the amount of traffic overhead due to key management in WSNs, and the computational cost is within acceptable limits.

Using a hardware/software codesign approach, a public key cryptosystem based on Rabin's scheme was implemented in motes developed by Tyndall National Institute [56]. It is claimed that the public key algorithms used minimal resources.

5.4 Key Management

Key management is a critically important problem in any type of cryptosystem. Key management involves: key storage, key agreement or distribution (among sensor nodes), and key maintenance (including revocation) [49]. In symmetric key cryptography, nodes must have a secure way to share secret keys before they can exchange data [22]. For example, there might be a network-wide shared key but there is obviously a risk that the key will be exposed if any node is compromised [4]. The key management problem is more complex and challenging in WSNs because of the resource limitations of nodes and the variety of security threats posed to nodes.

A taxonomy of symmetric key management in WSNs identifies three main approaches [6]:

- Base station participation: A trusted base station acts as a key distribution center (KDC) to create and give out unique link (session) keys.
- Trusted third node: A peer sensor node is a trusted intermediary that helps establish a shared key between node.
- Pre-distribution schemes using a master key, randomness, polynomials, matrix, tree, hierarchy, or combinatorics.

Many studies have focused on pre-distribution using randomness or polynomials. Random key pre-distribution basically focuses on the bootstrapping problem by issuing a different set of pre-established keys to each node, thereby reducing the probability that capturing one node will jeopardize the entire network. However, any two given nodes are not always guaranteed to be able to compute a pairwise key for a secure connection.

A symmetric key management scheme was implemented on a Silica board (Xynergy M4), which includes an ARM-M4 microprocessor and a Xilinx Spartan6 FPGA [11].

Traditionally, asymmetric key cryptography offers more elegant solutions to key distribution through the Diffie–Hellman key agreement protocol or public key infrastructure (PKI). The first question is suitability of Diffie–Hellman for WSNs. The Diffie–Hellman key agreement protocol appears at first glance to be infeasible for WSNs because it requires a key size of at least 1024 bits. However, ECC allows a much shorter key around 160 bits for the equivalent security level [4]. Therefore, the elliptic curve version of Diffie–Hellman has attracted a good deal of attention from researchers.

Elliptic curve Diffie–Hellman is part of a proposed key distribution protocol that establishes pairwise keys between nodes according to a specific routing algorithm, instead of loading full pairwise keys into each node [28]. Each node does not have to share a key with all neighbors except those involved in the routing path with it. This increases the resiliency against the threat of node capturing.

Kerberos and the elliptic curve Diffie–Hellman key exchange with the Elliptic Curve Digital Signature Algorithm (ECDH-ECDSA) were evaluated on MICAz and TelosB sensors [52]. Kerberos was found to be much less costly than ECDH-ECDSA

but has the major drawback that it requires a trusted third party. Given the ad hoc nature of WSNs, online central management is impractical [4].

Similarly, lightweight Kerberos was compared with ECMQV, an authenticated version of the elliptic curve Diffie–Hellman key exchange, in terms of energy cost [32]. Results showed that the ECMQV key exchange consumes up to twice as much energy as Kerberos-like key transport. Again, Kerberos is not well suited to WSNs because it is a centralized online approach.

Elliptic curve Diffie–Hellman was implemented in software on an 8-bit ATmega128L Micaz platform [39]. The implementation is shown to be not so demanding in flash memory space.

The next question is whether PKI is suitable for WSNs, having established that asymmetric key cryptography is feasible [49]. The functions of a PKI normally include registration, initialization, key generation, certification, certificate retrieval, and revocation. While WSNs are highly decentralized, base stations have been proposed to act as certificate authorities (CAs) and registration authorities (RAs). The base station generates the public–private key pair of a sensor node, assigns a unique identification to it, and creates the digital certificate that links the unique identification with its public key. It is generally believed that PKI is complex and probably too burdensome for WSNs. A possible solution is offered by the idea of identity-based cryptography (IBC), which has become a fertile area of research.

5.5 Identity-Based Cryptography

In theory, the problem of key management may be simplified considerably by IBC, first proposed by Shamir [76]. In IBC, public keys are derived from users' identities (e.g., name, email address, phone number). It eliminates the need to distribute public key certificates because an encrypted message can be sent to anyone without having to retrieve the recipient's public key. Shamir constructed an identity-based signature (IBS) scheme using RSA but was unsuccessful in constructing an identity-based encryption (IBE) scheme. In 2001, Boneh and Franklin came up with the first practical solution based on bilinear map pairings on elliptic curves [9]. Decryption depends on private keys distributed by a private key generator (PKG).

The security of IBC depends on the secrecy of the information stored in the PKG, so the PKG must be designed to be secure [67]. Good explanations of IBC for WSNs can be found in [67, 79].

The combination of IBC and ECC is particularly attractive because of the shorter keys made possible by ECC. Much of the research literature has focused on identity-based elliptic curve cryptography (IBE-ECC).

An identity-based elliptic curve cryptosystem based on Tate pairing, which is lightweight without any public key infrastructure and no key exchanges, was implemented on an Android phone platform [3]. It was shown to outperform existing IBE schemes in terms of complexity and efficiency.

A location-aware key authentication and distribution mechanism combining ECC and identity-based public key scheme was proposed [2]. In this scheme, public key authentication is based on the position of the sensor node in the monitored area. Before establishing a pairwise key between two nodes, each one of them must verify the neighborhood location of the other node using a message authentication code calculated on the corresponding public key using keys derived from encrypted beacons broadcast by anchor nodes.

An identity-based system is claimed to be secure against a variety of attacks including hello flood, wormhole, sinkhole, location deployment attack, man-in-the-middle attack, and masquerading as neighboring node [57].

IBC was used in a proposed heterogeneous online/offline signcryption scheme implemented in a Raspberry Pi B [81]. The proposed scheme was compared to four previous heterogeneous signcryption schemes.

Another identity-based public key scheme called C4W was proposed for certificate-less mutual authentication and key agreement [37]. Experiments showed savings in energy compared to simplified SSL (SSSL) protocol using an abbreviated certificate, but unfortunately no comparisons were made to other identity-based schemes.

6 Conclusions and Open Issues

Cryptography is the mathematical foundation for all secure protocols. It is a well-developed field with strong symmetric key ciphers such as AES, the well-known Diffie–Hellman key agreement protocol, and proven asymmetric key ciphers such as RSA routinely used for digital signatures, public key certificates, key distribution, and authentication. However, the implementation of cryptographic primitives in WSNs is not straightforward due to the unique constraints on energy, processing, memory, and bandwidth.

While symmetric key cryptography is generally preferred, studies have shown that asymmetric key cryptography, and in particular ECC, can also be feasible. Efficient implementation of elliptic curve cryptography (ECC) for WSNs is still a very active research topic, and techniques to further reduce the time and energy cost of ECC are eagerly sought.

Key management in WSNs remains an open research issue. A wide variety of symmetric key management schemes have been proposed without a solution that is clearly best.

In terms of asymmetric key management, the approach of elliptic curve Diffie–Hellman has been shown to be promising through a substantial amount of experimental work. As an alternative to PKI, researchers are looking at IBC and particularly the combination of IBC and ECC.

Another open issue is how cryptography can help data aggregation in the face of possibly malicious nodes. Not much work has been done in this area yet. A simple and provably secure encryption scheme that allows efficient additive aggregation

of encrypted data was proposed [12]. Only one modular addition is necessary for ciphertext aggregation. The security of the scheme is based on the indistinguishability property of a pseudorandom function. Chan did a formal treatment of concealed data aggregation (CDA) and the more general private data aggregation (PDA) [15]. Engouang described an ECC scheme with homomorphic properties allowing users to execute operations on encrypted metrics values [29]. Another data aggregation scheme called SA-SPKC uses stateful public key encryption and an additive homomorphic encryption and aggregate MAC to provide the end-to-end confidentiality and the end-to-end integrity [71].

References

1. Kerberos: The network authentication protocol. http://web.mit.edu/kerberos/
2. Abdallah, W., Boudriga, N.: A location-aware authentication and key management scheme for wireless sensor networks. In: 22nd Asia-Pacific Conference on Communications (APCC 2016), pp. 488–495. IEEE (2016)
3. Adiga, B.S., Rajan, M.A., Shastry, R., Shivraj, V.L., Balamuralidhar, P.: Lightweight IBE scheme for wireless sensor nodes. In: 2013 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), pp. 1–6. IEEE (2013)
4. Arazi, B., Elhanany, I., Arazi, O., Qi, H.: Revisiting public-key cryptography for wireless sensor networks. Computer **38**(11), 103–105 (2005)
5. Bala, S., Sharma, G., Verma, A.K.: Optimized elliptic curve cryptography for wireless sensor networks. In: 2nd IEEE International Conference on Parallel Distributed and Grid Computing (PDGC), pp. 89–94 (2012)
6. Bala, S., Sharma, G., Verma, A.K.: A survey and taxonomy of symmetric key management schemes for wireless sensor networks. In: CUBE International Information Technology Conference (CUBE 2012), pp. 585–592. ACM (2012)
7. Batina, L., Mentens, N., Sakiyama, K., Preneel, B., Verbauwhede, I.: Low-cost elliptic curve cryptography for wireless sensor networks. In: 2006 European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS'06) (2006)
8. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. J. Cryptol. **4**(1), 3–72 (1991)
9. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: CRYPTO 2001, pp. 213–229 (2001)
10. Burnett, S., Paine, S.: RSA Security's Official Guide to Cryptography. McGraw-Hill, New York, NY (2001)
11. Cardona, L.A., de la Fe, S., Lorente, B., Villar, S., Ferrer, C.: Secure key management in low power wireless sensor networks. In: 2013 47th International Carnahan Conference on Security Technology (ICCST), pp. 1–6. IEEE (2013)
12. Castelluccia, C., Chan, A.C.F., Mykletun, E., Tsudik, G.: Efficient and provably secure aggregation of encrypted data in wireless sensor networks. ACM Trans. Sens. Netw. **5**(3), 20:1–20:36 (2009)
13. Cayirci, E., Rong, C.: Security in Wireless Ad Hoc and Sensor Networks. Wiley, Chichester, UK (2009)
14. Chakavarika, T.T., Chaurasia, B.K., Gupta, S.K.: Performance evaluation of a polynomial based key management scheme in wireless sensor networks. In: 2016 International Conference on Communication and Signal Processing (ICCSP), pp. 2114–2118. IEEE (2016)
15. Chan, A.C.F., Castelluccia, C.: A security framework for privacy-preserving data aggregation in wireless sensor networks. ACM Trans. Sens. Netw. **7**(4), 29:1–29:45 (2011)

16. Chang, S.Y., Lin, Y.H., Sun, H.M., Wu, M.E.: Practical RSA signature scheme based on periodical rekeying for wireless sensor networks. ACM Trans. Sens. Netw. **8**(2), 13:1–13:13 (2012)
17. Choi, K.J., Song, J.I.: Investigation of feasible cryptographic algorithms for wireless sensor networks. In: International Conference on Advanced Computation and Telecommunication (ICACT'06) (2006)
18. Chu, D., Grosschadl, J., Liu, Z., Muller, V., Zhang, Y.: Twisted Edwards-form elliptic curve cryptography for 8-bit AVR-based sensor nodes. In: First ACM Workshop on Asia Public-key Cryptography (AsiaPKC 2013), pp. 39–44. ACM (2013)
19. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: RFC 5280, internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile (2008). https://tools.ietf.org/html/rfc5280
20. Coppersmith, D.: The data encryption standard (DES) and its strength against attacks. IBM J. Res. Dev. **38**(3), 243–250 (1994)
21. Cozzens, M., Miller, S.: The Mathematics of Encryption: An Elementary Introduction. American Mathematical Society, Providence, RI (2013)
22. Dargie, W., Poellabauer, C.: Fundamentals of Wireless Sensor Networks: Theory and Practice. Wiley, Chichester, UK (2010)
23. Dierks, T., Allen, C.: RFC 2246, the TLS protocol version 1.0 (1999). https://tools.ietf.org/html/rfc2246
24. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
25. Driessen, B., Poschmann, A., Paar, C.: Comparison of innovative signature algorithms for WSNs. In: First ACM Conference on Wireless Network Security (WiSec 2008), pp. 30–35. ACM (2008)
26. Du, X., Chen, H.H.: Security in wireless sensor networks. IEEE Wirel. Commun. **15**(4), 60–66 (2008)
27. Easttom, C.: Modern Cryptography: Applied Mathematics for Encryption and Information Security. McGraw-Hill Education, New York, NY (2016)
28. Eldefrawy, M.H., Khan, M.K., Alghathbar, K.: A key agreement algorithm with rekeying for wireless sensor networks using public key cryptography. In: 2010 International Conference on Anti-Counterfeiting, Security and Identification (ASID), pp. 1–6. IEEE (2010)
29. Engouang, T.D., Yun, L.: Aggregate over multi-hop homomorphic encrypted data in wireless sensor networks. In: 2013 2nd International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA), pp. 248–252. IEEE (2013)
30. Ganesan, P., Venugopalan, R., Peddabachagari, P., Dean, A., Mueller, F., Sichitiu, M.: Analyzing and modeling encryption overhead for sensor network nodes. In: 2003 ACM International Conference on Wireless Sensor Networks and Applications (WSNA'03) (2003)
31. Gaubatz, G., Kaps, J.P., Ozturk, E., Sunar, B.: State of the art in ultra-low power public key cryptography for wireless sensor networks. In: 3rd IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom 2005), pp. 146–150. IEEE (2005)
32. Grosschadl, J., Szekely, A., Tillich, S.: The energy cost of cryptographic key establishment in wireless sensor networks. In: 2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS 2007), pp. 380–382. ACM (2007)
33. Gura, N., Patel, A., Wander, A., Eberle, H., Shantz, S.: Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In: 2004 Workshop on Cryptographic Hardware and Embedded Systems, pp. 119–132 (2004)
34. Houssain, H., Badra, M., Al-Somani, T.F.: Hardware implementations of elliptic curve cryptography in wireless sensor networks. In: 6th International Conference for Internet Technology and Secured Transactions (ICITST 2011), pp. 1–6. IEEE (2011)
35. Huang, X., Sharma, D., Aseeri, M., Almorqi, S.: Secure wireless sensor networks with dynamic window for elliptic curve cryptography. In: 2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC), pp. 1–5. IEEE (2011)
36. ISO/IEC: ISO/IEC 9797-1:2011 information technology—security techniques—message authentication codes (MACs)—Part 1: Mechanisms using a block cipher. Technical report, ISO/IEC (2011). https://www.iso.org/standard/50375.html

37. Jing, Q., Hu, J., Chen, Z.: C4W: an energy efficient public key cryptosystem for large-scale wireless sensor networks. In: IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS), pp. 827–832. IEEE (2006)
38. Kerckhoffs, A.: La cryptographie militaire. J. Sci. Mil. **9**, 5–38 (1883). http://www.petitcolas.net/kerckhoffs/index.html
39. Khajuria, S., Tange, H.: Implementation of Diffie-Hellman key exchange on wireless sensor using elliptic curve cryptography. In: 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace and Electronic Systems Technology (Wireless VITAE 2009), pp. 772–776. IEEE (2009)
40. Koblitz, N.: Elliptic curve cryptosystems. Math. Comput. **48**, 203–209 (1987)
41. Koscielny, C., Kurkowski, M., Srebrny, M.: Modern Cryptography Primer: Theoretical Foundations and Practical Applications. Springer-Verlag, Berlin (2013)
42. Krawczyk, H., Bellare, M., Canetti, R.: RFC 2104, HMAC: keyed-hashing for message authentication (1997). https://tools.ietf.org/html/rfc2104
43. Krishnamachari, B.: Networking Wireless Sensors. Cambridge University Press, Cambridge, UK (2005)
44. Law, Y.W., Doumen, J., Hartel, P.: Survey and benchmark of block ciphers for wireless sensor networks. ACM Trans. Sens. Netw. **2**(1), 65–93 (2006)
45. Liu, A., Ning, P.: TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks. In: International Conference on Information Processing in Sensor Networks (IPSN'08), pp. 245–256. IEEE (2008)
46. Liu, D., Ning, P.: Security for Wireless Sensor Networks. Springer Science+Business Media, New York, NY (2007)
47. Liu, W., Luo, R., Yang, H.: Cryptography overhead evaluation and analysis for wireless sensor networks. In: WRI International Conference on Communications and Mobile Computing (CMC'09), pp. 496–501. IEEE (2009)
48. Liu, W., Ying, B., Yang, H., Wang, H.: Accurate modeling for predicting cryptography overheads on wireless sensor nodes. In: 11th International Conference on Advanced Communications Technology (ICACT 2009), vol. 2, pp. 997–1001. IEEE (2009)
49. Lopez, J., Zhou, J.: Wireless Sensor Network Security. IOS Press, Amsterdam (2008)
50. Merkle, R.: Secrecy, authentication, and public key systems. Ph.D. thesis, Stanford University (1979)
51. Merkle, R., Hellman, M.: On the security of multiple encryption. Commun. ACM **74**(24), 465–467 (1981)
52. de Meulenaer, G., Gosset, F., Standaert, F.X., Pereira, O.: On the energy cost of communication and cryptography in wireless sensor networks. In: 2008 IEEE International Conference on Wireless and Mobile Computing, Networking and communication (WIMOB'08), pp. 580–585. IEEE (2008)
53. Miller, V.S.: Use of elliptic curves in cryptography. In: Advances in Cryptology, CRYPTO'85, pp. 417–426. Springer-Verlag (1986)
54. Moh'd, A., Aslam, N., Phillips, W., Robertson, W., Marzi, H.: SN-SEC: a secure wireless sensor platform with hardware cryptographic primitives. Pers. Ubiquitous Comput. **17**(5), 1051–1059 (2013)
55. Mollin, R.: An Introduction to Cryptography, 2nd edn. Chapman and Hall/CRC, Boca Raton, FL (2007)
56. Murphy, G., Keeshan, A., Agarwal, R., Popovici, E.: Hardware-software implementation of public-key cryptography for wireless sensor networks. In: 2006 IET Irish Signals and Systems Conference, pp. 463–468. IET (2006)
57. Na, S.H., Kim, K.J., Hassan, M.M., Huh, E.N.: Identity-based secure protocol scheme for wireless sensor network. In: 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human (ICIS 2009), pp. 555–560. ACM (2009)
58. Nadir, I., Zegeye, W.K., Moazzami, F., Astatke, Y.: Establishing symmetric pairwise-keys using public-key cryptography in wireless sensor networks (WSN). In: IEEE 7th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), pp. 1–6. IEEE (2016)

59. Nair, B., Mala, C.: Analysis of ECC for application specific WSN security. In: IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), pp. 1–6. IEEE (2015)
60. Needham, R., Schroeder, M.: Using encryption for authentication in large networks of computers. Commun. ACM **21**(12), 993–999 (1978)
61. NIST: FIPS PUB 46-3 data encryption standard (DES). Technical report, NIST (1995). http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
62. NIST: FIPS PUB 197 advanced encryption standard (AES). Technical report, NIST (2001). http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
63. NIST: FIPS PUB 180-4 secure hash standard (SHS). Technical report, NIST (2015). http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
64. NIST: FIPS PUB 202 SHA-3 standard: Permutation-based hash and extendable-output functions. Technical report, NIST (2015). http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
65. Oreku, G., Pazynyuk, T.: Security in Wireless Sensor Networks. Springer International Publishing Switzerland, Cham (2016)
66. Patel, S.T., Mistry, N.H.: A survey: lightweight cryptography in WSN. In: 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005), pp. 11–15. IEEE (2005)
67. Patil, H.K., Szygenda, S.: Security for Wireless Sensor Networks Using Identity-Based Cryptography. CRC Press, Boca Raton, FL (2013)
68. Perrig, A., Stankovic, J., Wagner, D.: Security in wireless sensor networks. Commun. ACM **47**(6), 53–57 (2004)
69. Peterson, W.W., Brown, D.T.: Cyclic codes for error detection. Proc. IRE **49**(1), 228–235 (1961)
70. Piotrowski, K., Langendoerfer, P., Peter, S.: How public key cryptography influences wireless sensor node lifetime. In: 4th ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'06), pp. 169–176. ACM (2006)
71. Rafik, M.B.O., Mohammed, F.: SA-SPKC: secure and efficient aggregation scheme for wireless sensor networks using stateful public key cryptography. In: 11th International Symposium on Programming and Systems (ISPS), pp. 96–102. IEEE (2013)
72. Rijmen, V., Daemen, J.: The Design of Rijndael: AES–the Advanced Encryption Standard. Springer-Verlag, Berlin (2002)
73. Rivest, R.L.: RFC 1321, the MD5 message-digest algorithm, RFC 1321 (1992). https://tools.ietf.org/html/rfc1321
74. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
75. Roman, R., Alcaraz, C., Lopez, J.: A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes. Mobile Netw. Appl. **12**(4), 231–244 (2007)
76. Shamir, A.: Identity-based cryptosystems and signature schemes. In: CRYPTO 1984, LNCS, vol. 196, pp. 47–53. Springer-Verlag (1985)
77. Shannon, C.: Communication theory of secrecy systems. Bell Syst. Tech. J. **28**(4), 656–715 (1949)
78. Shim, K.A.: A survey of public-key cryptographic primitives in wireless sensor networks. IEEE Commun. Surv. Tutor. **18**(1), 577–601 (2016)
79. Sung, S., Ryou, J.: Id-based sensor node authentication for multi-layer sensor networks. J. Commun. Netw. **16**(4), 363–370 (2014)
80. Szczechowiak, P., Kargl, A., Scott, M., Collier, M.: On the application of pairing based cryptography to wireless sensor networks. In: 2nd ACM Conference on Wireless Network Security, pp. 1–12. ACM (2009)
81. Ting, P.Y., Tsai, J.L., Wu, T.S.: Signcryption method suitable for low-power IoT devices in a wireless sensor network. IEEE Syst. J. **PP**(99), 1–10 (2017)
82. Vamsi, P.R., Kant, K.: A taxonomy of key management schemes of wireless sensor networks. In: 5th International Conference on Advanced Computing and Communication Technologies (ACCT 2015), pp. 690–696. IEEE (2015)

83. Van Tilborg, H., Jajodia, S.: Encyclopedia of Cryptography and Security, 2nd edn. Springer, New York, NY (2011)
84. Wander, A., Gura, N., Everle, H., Gupta, V., Shantz, S.C.: Energy analysis of public-key cryptography for wireless sensor networks. In: 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005), pp. 324–328. IEEE (2005)
85. Watro, R., Kong, D., Cuti, S.-f., Gardiner, C., Lynn, C., Kruus, P.: TinyPK: securing sensor networks with public key technology. In: 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2004), pp. 59–64. ACM (2004)
86. Zhang, Y., Grosschadl, J.: Efficient prime-field arithmetic for elliptic curve cryptography on wireless sensor nodes. In: 2011 International Conference on Computer Science and Network Technology, pp. 459–466. IEEE (2011)
87. Zheng, J., Jamalipour, A.: Wireless Sensor Networks: A Networking Perspective. Wiley, Hoboken, NJ (2009)