Jorge Sá Silva
Bhaskar Krishnamachari
Fernando Boavida (Eds.)

LNCS 5970

Wireless Sensor Networks

7th European Conference, EWSN 2010
Coimbra, Portugal, February 2010
Proceedings

Springer

Lecture Notes in Computer Science 5970

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Jorge Sá Silva Bhaskar Krishnamachari
Fernando Boavida (Eds.)

Wireless Sensor Networks

7th European Conference, EWSN 2010
Coimbra, Portugal, February 17-19, 2010
Proceedings

Springer

Volume Editors

Jorge Sá Silva
University of Coimbra, Department of Informatics Engineering
Polo II, Pinhal de Marrocos, 3030-290 Coimbra, Portugal
E-mail: sasilva@dei.uc.pt

Bhaskar Krishnamachari
University of Southern California
Department of Electrical Engineering - Systems
3740 McClintock Avenue, EEB 300, Los Angeles, CA 90089, USA
E-mail: bkrishna@usc.edu

Fernando Boavida
University of Coimbra, Department of Informatics Engineering
Polo II, Pinhal de Marrocos, 3030-290, Coimbra, Portugal
E-mail: boavida@uc.pt

Library of Congress Control Number: 2010920237

CR Subject Classification (1998): C.2.4, C.2, F.2, D.1.3, D.2, E.1, H.4, C.3

LNCS Sublibrary: SL 5 – Computer Communication Networks
and Telecommunications

ISSN 0302-9743
ISBN-10 3-642-11916-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-11916-3 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12995295 06/3180 5 4 3 2 1 0

Preface

It is our great pleasure to present the proceedings of the European Conference on Wireless Sensor Networks 2010 (EWSN 2010).

As the field of wireless sensor networks matures, new design concepts, experimental and theoretical findings, and applications have continued to emerge at a rapid pace. As one of the leading international conferences in this area, EWSN has played a substantial role in the dissemination of innovative research ideas from researchers all over the globe.

EWSN 2010 was organized by the University of Coimbra, Portugal, during February 17–19, 2010 and it was the seventh meeting in this series. Previous events were held in Berlin (Germany) in 2004, Istanbul (Turkey) in 2005, Zurich (Switzerland) in 2006, Delft (The Netherlands) in 2007, and Cork (Ireland) in 2009.

A high-quality selection of papers made up EWSN 2010. Based on the reviews and the recommendations from the four live TPC discussions, we selected a total of 21 papers from 109 submissions (19.26% acceptance rate) for EWSN 2010. Topics of interest included hardware design and implementation, operating systems and software, middleware and macroprogramming, communication and network protocols, information and signal processing, fundamental theoretical limits and algorithms, prototypes, field experiments, testbeds, novel applications, including urban sensing, security and fault-tolerance.

Putting together EWSN 2010 was a team effort. We would like to thank the Program Committee members, the reviewers, our sponsors, all authors, and the Organizing Committee for their respective contributions.

We believe the conference program was interesting and that it provided participants with a very valuable opportunity to share ideas with other researchers and practitioners strongly involved in wireless sensor networks.

February 2010

Bhaskar Krishnamachari
Fernando Boavida
Jorge Sá Silva

Organization

General Co-chairs

Jorge Sá Silva	University of Coimbra, Portugal
Fernando Boavida	University of Coimbra, Portugal

TPC Co-chairs

Bhaskar Krishnamachari	University of Southern California, USA
Jorge Sá Silva	University of Coimbra, Portugal

Program Committee

Adam Dunkels	SICS, Sweden
Adam Wolitz	Technical University of Berlin, Germany
Alex Dimakis	University of Southern California, USA
Alexander Pflaum	Fraunhofer Institute, Germany
Andreas Terzis	Johns Hopkins University, USA
Andreas Willig	Technical University of Berlin, Germany
Attila Vidacs	Budapest Universtity of Technology and Economics, Hungary
Bjorn Pehrson	KTH, Sweden
Chen Avin	Ben-Gurion University of the Negev, Israel
Cláudio Geyer	FRGS University, Brazil
Cormac Sreenan	University College Cork, Ireland
Edmundo Monteiro	University of Coimbra, Portugal
Eduardo Nakamura	FUCAPI, Brazil
Hannes Frey	University of Paderborn, Germany
Holger Karl	University of Paderborn, Germany
Jan Beutel	ETH Zurich, Switzerland
Jaudelice Oliveira	Drexel University, USA
Jie Gao	Stony Brook, USA
Joe Polastre	Sentilla, USA
Kamin Whitehouse	University of Virginia, USA
Kasun De Zoysa	University of Colombo, Sri Lanka
Koen Langendoen	Delft University of Technology, The Netherlands
Lars Wolf	TU Braunschweig, Germany
Lin Zhang	Tsinghua University, China
Luca Mottola	SICS, Sweden
Manuel Ricardo	INESC Porto, Portugal
Marimuthu Palaniswami	University of Melbourne, Australia
Mário Alves	Polytechnic Institute of Porto, Portugal

Martin Haenggi	Notre Dame, USA
Matthias Hollick	Universidad Carlos III de Madrid, Spain
Matt Welsh	Harvard University, USA
Michele Zorzi	University of Padova, Italy
Neeli Prasad	Aalborg University, Denmark
Ozlem Durmaz-Incel	Bogazici University, Turkey
Paul Havinga	University of Twente, The Netherlands
Pedro Marron	University of Bonn and Fraunhofer IAIS, Germany
Rolland Vida	Budapest University of Technology and Economics, Hungary
Rui Rocha	IT, Portugal
S. Mukhopadhyay	Massey University, New Zealand
Sanjay Jha	UNSW, Australia
Suman Nath	Microsoft Research, USA
Torsten Braun	University of Bern, Switzerland
Utz Roedig	Lancaster University, UK
V.S. Anil Kumar	VirginiaTech, USA
Wendi Heinzelman	University of Rochester, USA
Yu Chen	State University of New York – Binghamton, USA

Tutorial Co-chairs

Andreas Terzis	Johns Hopkins University, USA
Joel Rodrigues	University of Beira Interior IT, Portugal

Poster and Demo Co-chairs

Paulo Pinto	UNL, Portugal
Slaven Marusic	University of Melbourne, Australia

Publicity Co-chairs

Fernando Velez	IT, Portugal
Pei Zhang	CMU, USA
Takahiro Hara	Osaka University, Japan

Sponsorships Co-chairs

Marília Curado	University of Coimbra, Portugal
Vasos Vassiliou	University of Cyprus, Cyprus

Publication Chair

Pedro Furtado	University of Coimbra, Portugal

Local Arrangements Chair

Paulo Simões	University of Coimbra, Portugal

Local Arrangements Committee

Alberto Cardoso	University of Coimbra, Portugal
André Rodrigues	University of Coimbra, Portugal
Jorge Granjal	University of Coimbra, Portugal
Laura Peralta	University of Madeira, Portugal
Milan Simek	University of Brno, Czech Republic
Paulo Gil	University of Coimbra, Portugal
Ricardo Silva	University of Coimbra, Portugal
Vasco Pereira	University of Coimbra, Portugal

Sponsors

Gold: CONET
Silver: Eneida, Fundação Luso-Americana, Libelium
Standard: Galp

Table of Contents

Localization, Synchronization and Compression

Networking – I

New Directions

Programming & Architecture

Link Reliability

Networking – II

Radio Interferometric Angle of Arrival Estimation

Isaac Amundson, Janos Sallai, Xenofon Koutsoukos, and Akos Ledeczi

Institute for Software Integrated Systems (ISIS)
Department of Electrical Engineering and Computer Science
Vanderbilt University
Nashville, TN 37235, USA
isaac.amundson@vanderbilt.edu

Abstract. Several localization algorithms exist for wireless sensor networks that use angle of arrival measurements to estimate node position. However, there are limited options for actually obtaining the angle of arrival using resource-constrained devices. In this paper, we describe a radio interferometric technique for determining bearings from an anchor node to any number of target nodes at unknown positions. The underlying idea is to group three of the four nodes that participate in a typical radio interferometric measurement together to form an antenna array. Two of the nodes transmit pure sinusoids at close frequencies that interfere to generate a low-frequency beat signal. The phase difference of the measured signal between the third array node and the target node constrains the position of the latter to a hyperbola. The bearing of the node can be estimated by the asymptote of the hyperbola. The bearing estimation is carried out by the node itself, hence the method is distributed, scalable and fast. Furthermore, this technique does not require modification of the mote hardware because it relies only on the radio. Experimental results demonstrate that our approach can estimate node bearings with an accuracy of approximately 3° in 0.5 sec.

1 Introduction

Spatial coordination in wireless sensor networks (WSNs) has received a lot of attention in recent years. In typical solutions, one or more nodes emit a signal, and some property of that signal (e.g. angle of arrival (AOA), time of arrival (TOA), received signal strength (RSS), etc.) is measured and used to derive bearing or range. Angulation or lateration techniques can then respectively be used to estimate a node's position.

Although several techniques exist for determining node position based on bearing information [1], [2], [3], [4], [5], there are few options for actually measuring signal AOA in WSNs. Currently available methods for bearing estimation require a heavy-weight infrastructure [6], rotating hardware [7], [8], directional antennas [9], and/or expensive and sophisticated sensors [10]. Furthermore, such techniques typically require participating nodes to be stationary for extended periods of time. These constraints are often undesirable for WSN deployments, in

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 1–16, 2010.

which node size and cost must be kept to a minimum. An AOA approach that does not require additional hardware, runs on the nodes themselves, and is fast enough to support tracking in addition to static localization would be a major step forward.

In this paper, we propose a novel AOA approach for WSNs that uses radio interferometry [11]. The basic idea is to group together three of the four nodes involved in a typical radio interferometric measurement to form an antenna array, which acts as an anchor node. Two transmitters and one receiver are arranged in such a manner that their antennas are mutually orthogonal to minimize parasitic antenna effects (see Figure 1.) The measured phase difference between the receiver in the array and a target node constrains the location of the latter to a hyperbola. The bearing of the target node can then be estimated by computing the angle of the hyperbola asymptote, assuming the target node is not too close to the array.

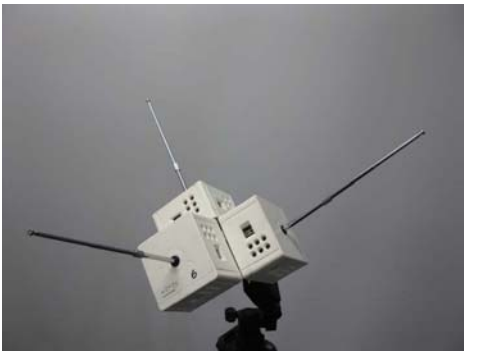


Fig. 1. Antenna array implementation using three XSM motes

We present several new contributions for estimating the angle of arrival in wireless sensor networks.

1. We describe an RF-based technique for determining target node bearing.
2. We provide a detailed analysis that shows our bearing estimation algorithm is robust to measurement noise and approximation error.
3. We design a real-world implementation using COTS sensor nodes, in which bearing estimation is performed entirely on the resource-constrained motes.
4. We present experimental results that show our approach can rapidly and accurately estimate node bearing.

The remainder of this paper is organized as follows. In Section 2, we discuss other angle of arrival techniques for WSNs. Section 3 describes our proposed system, followed by an error analysis in Section 4. In Section 5, we describe our implementation on a real-world WSN platform. In Section 6, we evaluate our system based on experimental results. Section 7 concludes.

2 Related Work

The RF method we use for determining AOA is based on radio interferometry. The Radio Interferometric Positioning System (RIPS) provides accurate RF-based localization in WSNs [11]. The main idea is that the resource-constrained nodes cannot sample a pure RF signal fast enough, but can process the lower-frequency envelope of the beat signal that results from the interference of two high-frequency signals. The difference in signal phase measured by two other nodes is a linear combination of the distances between the transmitters and receivers, modulo the wavelength, and can be used for localizing all participating nodes by solving an optimization problem. Although RIPS has centimeter-accuracy and can support inter-node distances of greater than twice the communication range, it requires centralized processing, suffers from high latency, and involves sampling at several frequencies.

A broad spectrum of acoustic beamforming techniques have been proposed to find the angle of incidence of a signal at an array of sensors. The most common techniques include delay-and-sum beamforming, Capon beamforming [12], MUSIC [13], ESPRIT [14] and min-norm [15] algorithms. Since the time of flight of the signal from the source to sensors in the array varies based on their pairwise distances, sensors receive the signal with different phases. While all of these methods compute the bearing of the source from the data streams sampled at the individual sensors, they differ greatly with respect to their angular resolution as well as their computational requirements. In WSNs, angular resolution is typically within 10° [16].

The Cricket Compass [17] is a device which uses ultrasound to determine orientation with respect to a number of ceiling-mounted beacons. Two receivers are mounted a few centimeters apart on a portable device, and the phase difference of the ultrasonic signal is measured to determine bearing. Although both the Cricket Compass and our approach measure signal phase difference to derive AOA, the two systems use different hardware, signal modalities, phase disambiguation techniques, and bearing derivation algorithms. The Cricket Compass has an accuracy of between 3° and 5°, depending on the orientation of the compass.

Angle of arrival can be used in different ways for spatial coordination. Triangulation, for example, is the process of determining the position of an object from the bearings of known reference positions. Two such reference positions (or three non-collinear ones in degenerate cases) are enough to localize any number of nodes within range. In [2], a method is given to determine position based on the angular separation (the difference in bearings) between beacons. Other angle of arrival positioning approaches have been developed, including multiangulation using subspace methods [4], anchor bearing propagation [1], and semidefinite programming [3]. Bearing estimates can also be useful when anchor positions are unknown. In [18] and [19], mobile robot navigation methods are presented for arriving at a target position by only observing angular separation between two pairs of landmarks.

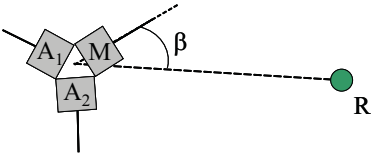


Fig. 2. Array containing a master node (M) and two assistant nodes (A_1, A_2). A target node (R) computes its bearing (β) from the array.

3 System Overview

Radio Interferometric Measurements. Our system consists of a stationary antenna array and cooperating wireless sensor nodes at unknown positions. We assume that the position of the midpoint of the array is known, as well as the distance between the antennas in the array. The array contains three nodes, a master (M) and two assistants (A_1, A_2), as shown in Figure 2. At a predetermined time, the master, M, and one of the assistants, A_1, transmit a pure sinusoidal signal at slightly different frequencies, which interfere to create a low-frequency beat signal whose phase is measured by the other assistant in the array, A_2, and a receiver node, R, at an unknown position. Such a measurement is termed a radio interferometric measurement (RIM).

The difference in phase, $\Delta\varphi = \varphi_R - \varphi_{A_2}$, measured by receiver nodes R and A_2 is a linear combination of the distances between the transmitters and receivers,

$$\Delta\varphi = \frac{2\pi}{\lambda}(d_{MA_2} - d_{A_1A_2} + d_{A_1R} - d_{MR}) \ (mod\ 2\pi),$$

where λ is the wavelength of the carrier frequency, d_{MR} is the distance between the master node and target receiver node, d_{A_1R} is the distance between the assistant transmitter and the target receiver node, and d_{MA_1}, d_{MA_2}, and $d_{A_1A_2}$ are the respective distances between all pairs of nodes in the array. Note that the nodes in the array are equidistant from each other, and therefore $d_{MA_2} - d_{A_1A_2} = 0$, so the phase difference can be simplified:

$$\Delta\varphi = \frac{2\pi}{\lambda}(d_{A_1R} - d_{MR}) \ (mod\ 2\pi). \tag{1}$$

We denote the distance difference $d_{A_1R} - d_{MR}$ by d_{A_1MR} and refer to it as a *t-range*. From Equation (1), we can see that if $-\frac{\lambda}{2} < d_{A_1MR} < \frac{\lambda}{2}$, the phase difference will fall in the interval $(-\pi, \pi)$. When this is *not* the case, the possible range of $\Delta\varphi$ will exceed 2π, which results in a modulo 2π phase ambiguity. To avoid this, we would like the maximum possible distance difference to be less than $\frac{\lambda}{2}$. The maximum distance difference will occur when the receiver node is collinear with the transmitters M and A_1. d_{A_1MR} then corresponds to the distance between the master and assistant. Therefore, to eliminate the modulo 2π phase ambiguity, we require the distance between antennas in the array to be less than half the wavelength of the carrier frequency.

Having removed the modulo operator, we can rearrange Equation (1) so that known values are on the right hand side.

$$d_{A_1MR} = \frac{\Delta\varphi\lambda}{2\pi} \tag{2}$$

The t-range d_{A_1MR} defines an arm of a hyperbola that intersects the position of node R, and whose asymptote passes through the midpoint of the line $\overline{A_1M}$, connecting the master and assistant nodes. Figure 3 illustrates such a hyperbola with foci A_1 and M. The absolute value of the distance differences between the foci and any point on a hyperbolic arm is constant, formally defined as

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$$

where (x, y) are the coordinates of a point on the hyperbola, a is the distance between the hyperbola center and the intersection H of the hyperbola with the axis connecting the two foci, and b is the length of the line segment, perpendicular to the axis connecting the foci, that extends from H to the asymptote.

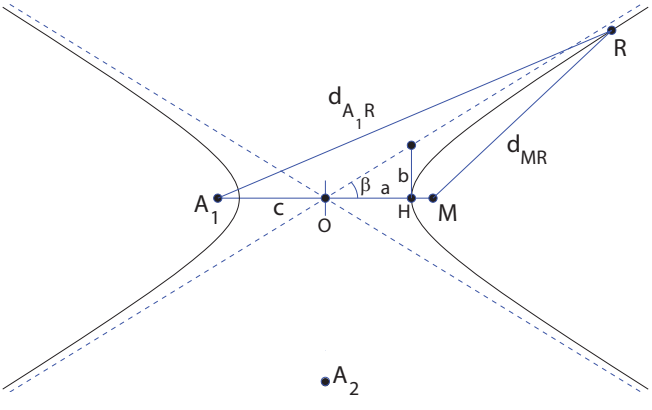


Fig. 3. The t-range defines a hyperbola that intersects node R, and whose asymptote passes through the midpoint of the two transmitters in the array.

Bearing Approximation. The hyperbola in Figure 3 is centered at O, and the distance between O and either focus is denoted by c. Furthermore, it can be shown that $c^2 = a^2 + b^2$ [20]. From the figure, we see that the bearing of the asymptote is $\beta = tan^{-1}(\frac{b}{a})$. Therefore, in order to solve for β, we must determine the values of b and a.

We can solve for a by observing that

$$d_{A_1R} - d_{MR} = d_{A_1H} - d_{MH}$$

because, by definition, the distance differences between the foci and all points on the hyperbola are constant. From Figure 3, we see that we can substitute $(c + a)$ for d_{A_1H} and $(c - a)$ for d_{MH}, and therefore,

$$d_{A_1R} - d_{MR} = (c + a) - (c - a) = 2a.$$

From Equation (2), we know the value of $d_{A_1R} - d_{MR}$, which is the t-range, and therefore $a = \frac{d_{A_1MR}}{2}$. We can then solve for b, using $b = \sqrt{c^2 - a^2}$. In terms of known distances, the bearing of the asymptote is then defined as

$$\beta = tan^{-1}\left(\frac{\sqrt{\left(\frac{d_{A_1M}}{2}\right)^2 - \left(\frac{d_{A_1MR}}{2}\right)^2}}{\left(\frac{d_{A_1MR}}{2}\right)}\right). \tag{3}$$

In Figure 3, we see the case where $d_{A_1MR} > 0$, and the position of R lies on the right arm of the hyperbola. If the phase difference is negative (i.e., $\varphi_R < \varphi_{A_2}$) then the position of R will lie on the left arm of the hyperbola. When this is the case, β is taken clockwise, and we must adjust it by subtracting it from π.

The line $\overline{A_1M}$ connecting the two foci is called the transverse axis of the hyperbola, and is a line of symmetry. This implies that although we know b, we do not know its sign, because mirrored positions on either side of the transverse axis will result in the same d_{A_1MR}. Therefore, the asymptote bearing β we obtained using this method could be either positive or negative. To find which bearing is correct, we can switch the roles of the assistant nodes in the array and perform another RIM. This will generate a different t-range, and hence another hyperbolic arm with foci A_2 and M.

Each hyperbola provides us with two angles $\pm\beta_i$, where β_i is the angle of the asymptote with the transverse axis, $\overline{A_iM}$. Of course, these angles will be offset from the global x-axis, because the orientation of $\overline{A_iM}$ may not be 0. Adjusting for this, one of the β_1 bearings, and one of the β_2 bearings will point in the same direction, which will approximate the actual bearing of R, as illustrated in Figure 4. Due to the position difference between the centers of the two hyperbolas, we do not expect these two angles to be equal, therefore we define a small

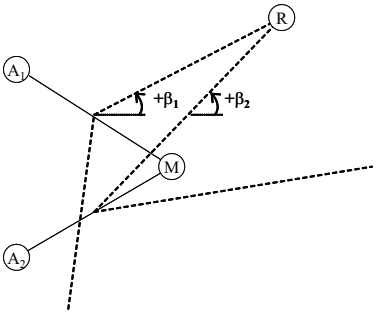


Fig. 4. Determining the true bearing of R is accomplished by selecting $+\beta$ or $-\beta$ from each master-assistant pair, such that the difference between the two angles is below the threshold ϵ_β

threshold ϵ_β, such that if $|\beta_1 - \beta_2| < \epsilon_\beta$, these two angles are considered a match. We then take the average of the two angles to obtain our bearing estimate, $\hat{\beta}$.

Because points on the hyperbola converge with the asymptote as their distance from the hyperbola center increases, the bearing approximation error is larger when R is close to the array. We therefore make the assumption that node R is a sufficient distance from the array. In Section 4, we show that this distance does not need to be large when using small-aperture arrays.

4 Error Analysis

In this section, we present an error analysis of the proposed bearing estimation technique. It is important to note that, although we use phase differences as input to our bearing estimation algorithm, the algorithm is generalizable to small-aperture sensor arrays that can derive distance differences using *any* means. For instance, RF ultra wide band antenna arrays, acoustic or ultrasonic sensors, and other types of arrays that can yield time-difference-of-arrival (TDOA) measurements from (sufficiently) distant sources fall into this category. Therefore, in this section, we assume the inputs to be distance differences. Notice that the distance differences are linearly related to RIM measurements (see Equation (2)), and therefore the error sensitivity results presented below remain valid. In the generalized case, the same applies to TDOA measurements, from which the distance differences can be computed via multiplication of the respective signal propagation speed (speed of sound for acoustic, speed of light for RF).

Typically, bearings are computed from distance differences by solving a non-linear set of equations using iterative techniques. Such techniques are prohibitive on low-end microprocessors due to their computational cost. We make a set of assumptions that allows us to compute bearing estimates in a reasonably simple way. While our bearing estimation technique is computationally less expensive than traditional nonlinear optimization techniques, our simplifying assumptions introduce estimation errors, which we identify below.

- *Measurement noise.* The distance differences observed by the receiver nodes contain measurement noise. The measurement noise can be attributed to, for instance, non-ideal signal propagation, noise from the electrical circuitry of the receiver, sampling error and quantization error of the analog-to-digital converter (ADC).
- *Asymptote approximation.* For a pair of transmitters, we approximate the bearing of the receiver with the angles of the asymptotes of the hyperbola. This is a good approximation if the receiver is sufficiently far from the transmitters, because the hyperbola converges on its asymptote. However, for close receivers, errors due to this assumption will not be negligible.
- *Translation of bearing candidates.* At least two transmitter pairs are required to unambiguously compute the bearing because, for just one transmitter pair, the angles of both asymptotes are possible solutions. We refer to the two solutions as *bearing candidates*. Since, for a transmitter pair, we compute the bearing candidates with respect to the midpoint of the segment defined by

the two antennas, fusing bearing candidates from two different transmitter pairs is not possible without knowing the distance of the receiver. We use the far-field assumption (i.e., that the receiver is infinitely far from the transmitter array) to carry out the disambiguation and fusion of bearings, introducing an error this way.

We intentionally omit the analysis of array position and orientation errors and instead make the following assumptions:

- *Antenna configuration is known.* The transmitter locations are assumed to be given. It is assumed that the transmitter nodes are fabricated with a prescribed antenna separation.
- *Relative bearings.* We assume that the computed bearings are given in the local coordinate system of the array. Hence, the location and orientation errors of the array are not considered in the error analysis of the bearing estimation.

We first analyze the sensitivity of the bearing estimates to noise in the distance difference inputs. Second, we analytically derive the errors related to the asymptote approximation and to the translation of bearing candidates. These errors depend on the bearing and distance of the target receiver, relative to the transmitter array. Finally, we provide an analysis of the total bearing estimation error resulting from both noise in the inputs and the errors due to the asymptote approximation and the translation of bearing candidates.

Sensitivity of bearing to measurement noise. A distance difference from a pair of transmitters in the array constrains the location of the receiver to one arm of a hyperbola, the foci of which are the positions of the two transmitters. For the sake of simplicity, let us assume that the two transmitters M and A_1 are located at $(c, 0)$ and $(-c, 0)$, respectively (see Figure 3). If the measured distance difference is positive, the receiver is constrained to the right arm of the hyperbola, while if the distance difference is negative, the receiver is located on the left arm. We approximate the bearing of the receiver using the asymptote angles as follows:

$$\hat{\beta} = \begin{cases} \pm tan^{-1}\left(\frac{\sqrt{c^2-a^2}}{a}\right), & \text{if } a > 0 \\ \pm\frac{\pi}{2}, & \text{if } a = 0 \\ \pi \mp tan^{-1}\left(\frac{\sqrt{c^2-a^2}}{a}\right), & \text{if } a < 0 \end{cases} \tag{4}$$

We analyze the sensitivity of the bearing estimates $\hat{\beta}$ to noise in the distance difference by taking the partial derivative of Equation (4) with respect to the distance difference. To see what amplification effect an error in a given distance difference d produces on the bearing estimate, we need to evaluate the partial derivative at d.

Figure 5a shows the relation between the measured distance difference d and the bearing candidates $\hat{\beta}$ when the antenna separation is half the wavelength

($\frac{\lambda}{2}$). Notice the ambiguity of the bearing candidates. Figure 5b plots $\frac{\delta\hat{\beta}}{\delta d}$ for each solution of $\hat{\beta}$. This figure shows that when the absolute value of the measured distance difference is close to the antenna separation, the computed bearing candidates are very sensitive to measurement noise. For instance, if the distance difference is 80% of the antenna separation, an infinitesimally small error in the measurement will be amplified ten-fold in the bearing estimate.

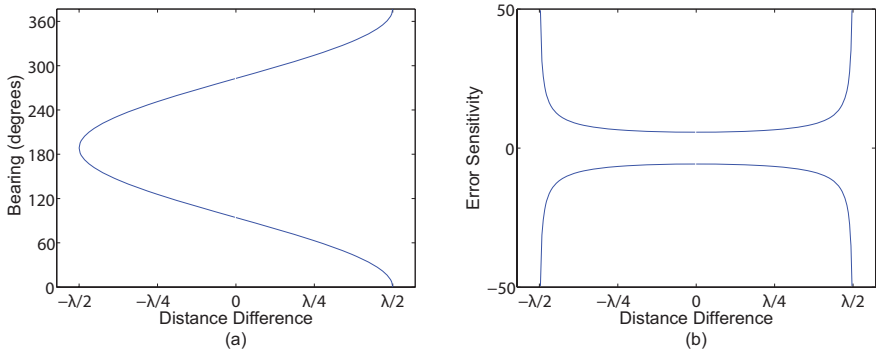


Fig. 5. (a) Relationship between measured distance difference and computed bearing. (b) Sensitivity of the computed bearing to measurement noise.

Asymptote approximation. The error of approximating a hyperbola with its asymptote is the difference between the approximated bearing $\hat{\beta}$ and the actual bearing β of the receiver. Assuming that the receiver R is located at (u, v), the actual bearing will be $\beta = tan^{-1}(\frac{v}{u})$. Hence, the error ϵ introduced by the asymptote assumption is

$$\epsilon = \hat{\beta} - \beta = \begin{cases} \pm tan^{-1}\left(\frac{v}{u}\right) \mp tan^{-1}\left(\frac{\sqrt{c^2-a^2}}{a}\right), & \text{if } a > 0 \\ 0, & \text{if } a = 0 \\ \mp tan^{-1}\left(\frac{v}{u}\right) \pm tan^{-1}\left(\frac{\sqrt{c^2-a^2}}{a}\right), & \text{if } a < 0 \end{cases} \quad (5)$$

Figure 6a shows the error introduced by the asymptote approximation when the receiver is located respectively one, two, and three times the antenna distance away from the midpoint of the segment connecting the two antennas. As expected, the error of the approximation decreases as the distance of the receiver from the transmitter array increases, that is, as the hyperbola converges on its asymptote. As we can see, the maximum error introduced by the asymptote assumption is less than 0.6°, as little as three antenna distances away.

Translation of bearing candidates. For a pair of transmitter antennas, it is not possible to unambiguously approximate the bearing of the asymptote. Because the hyperbola arm has two asymptotes, the angle of either one can be the correct bearing estimate. Hence, we need two transmitter antenna pairs for disambiguation. Let us treat the bearing candidates (computed from the t-ranges

of both transmitter antenna pairs) as vectors of unit length, with bases at the center of the hyperbolas, and whose angles are the computed bearing candidates. Since these vectors are given in the coordinate system of the respective hyperbolas, we need to transform them to the coordinate system of the array. This transformation includes a translation and a rotation. Then, we translate each vector such that its base is at the origin. Clearly, the bearing vector translated this way will not point directly toward the target receiver anymore, but if the receiver is sufficiently far from the transmitter array, the introduced angular error will be small. Finally, we disambiguate the bearing candidates by finding two that have approximately the same value.

Let us now express the angular error caused by the translation of bearing candidates. We assume that the transmitter is a uniform circular array of three antennas, with pairwise antenna distance of $\frac{\lambda}{2}$. The coordinate system of the array is set up such that the midpoint of the array is at the origin, and antenna M lies on the positive side of the x-axis. Let us consider only the correct bearing candidate (the other will be discarded later) for transmitter pair M and A_1. Furthermore, let us assume for now that the bearing candidate has no error. The difference between the actual bearing of the target receiver and the angle of the bearing candidate translated to the origin gives the angular error of the far-field assumption.

Figure 6b shows the error introduced by the far-field assumption when the receiver is located respectively one, two, and three times the antenna distance away from the midpoint of the segment connecting the two antennas. As we can see, as few as three antenna distances away, the maximum error introduced by the antenna assumption is less than 5°. In this particular antenna arrangement, the maximum errors are at 15° and 225°, respectively.

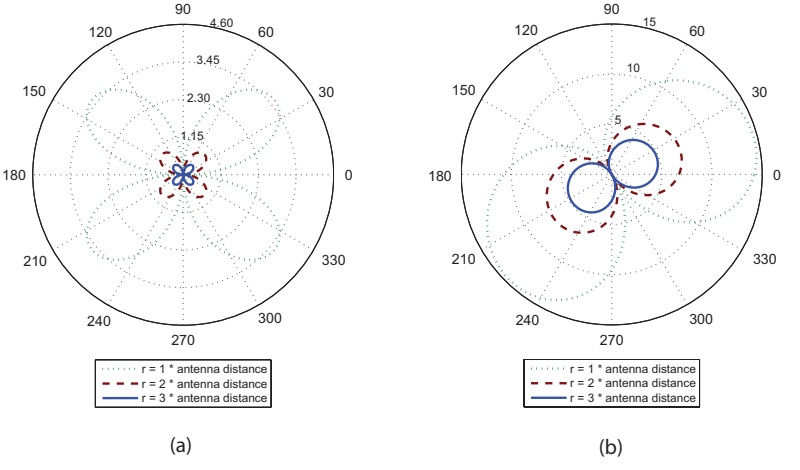


Fig. 6. Error in bearing (in degrees) caused by (a) the assumption that the receiver lies on the asymptote, and (b) assuming that bearing from the midpoint of the segment connecting the two antennas equals the bearing from the origin of the array coordinate system

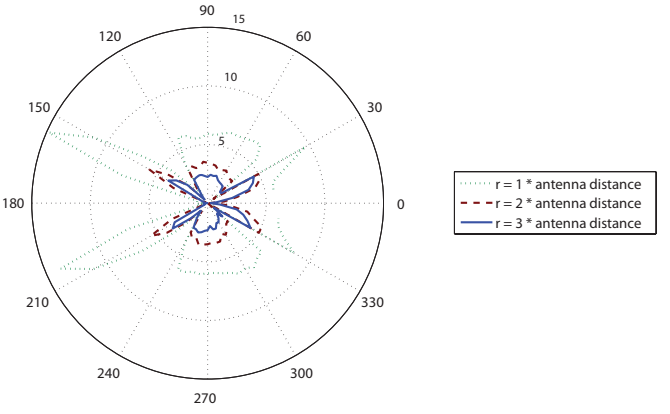


Fig. 7. Absolute error of bearing estimation (in degrees) caused by noisy distance differences, averaged over 500 simulation rounds. The standard deviation of the distance difference errors is 5% of the antenna distance.

Compound bearing estimation error. Since one transmitter pair reports two bearing candidates, at least two transmitter pairs are required to resolve this ambiguity. For the sake of simplicity, let us assume that we have two transmitter pairs. Clearly, there must be one bearing candidate for each transmitter pair that is close to the true bearing. Except for some degenerate cases, the other two bearing candidates will be significantly different than the true bearing, and will not be close to each other (see Figure 4). Therefore, in order to disambiguate between the correct and incorrect bearing candidates, we take all possible pairs of bearing candidates, one from the first transmitter pair and the other from the second transmitter pair, and find the pair with the least pairwise angular difference. The reported bearing estimate is computed as the average of the two closest bearing candidates.

Figure 7 shows the bearing estimation errors considering the above three types of error sources, averaged over 500 simulation rounds. We added a Gaussian noise to the distance differences, with mean zero and standard deviation set to 5% of the antenna distance. The plot suggests that the expected bearing estimation errors are below 5°, and peak around 30°, 150°, 240° and 330°, exactly where the individual transmitter pairs exhibit high error sensitivity.

5 Implementation

Our system is implemented using Crossbow ExScal motes (XSMs) [21], which use the Texas Instruments CC1000 radio chip and transmit in the 433 MHz range. Three XSMs form the array. Because the two transmitting antennas are close to each other, they will suffer from parasitic effects [22]. To minimize this negative interference, we place the nodes in a mutually orthogonal configuration.

All sensor nodes are elevated approximately 1.5 meters to reduce ground-based reflections. The antenna array is pictured in Figure 1. All nodes in our system execute the same distributed application, coded in nesC, and run the TinyOS operating system. All operations run locally, and there is no offline or PC-based processing involved. The entire application requires 3 kilobytes of RAM and 55 kilobytes of program memory (ROM).

Run-time. Figure 8 is a sequence diagram of the system run-time components using a setup of one array and a single target receiver node. Because phase difference is used to determine bearing, each node must measure the signal phase at the same time instant. This requires synchronization with accuracy on the order of microseconds or better. A SyncEvent message [23] is broadcast by the master transmitter, and contains a time in the future for all participating nodes to start the first RIM. Each array then performs two RIMs, one for each master-assistant pair. Signal transmission involves acquiring and calibrating the radio, transmitting the signal, then restoring the radio to enable data communication. The assistant nodes in the array store their phase measurements until both master-assistant pairs have finished their RIMs, at which point they broadcast their phase measurements to the target nodes. The target nodes then calculate their bearings from the array.

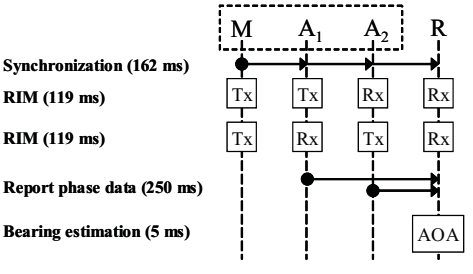


Fig. 8. Sequence diagram of RIM schedule with one array (dotted box) and the target receiver node (R)

6 Evaluation

To evaluate the accuracy of our system, we perform two experiments. In Experiment 1, we measure the bearing accuracy of six receiver nodes, which are evenly spaced around the array every 60° at a distance of ten meters from the array center. This experiment demonstrates how the bearing error changes with respect to array orientation. In Experiment 2, we measure the bearing accuracy of 14 receiver nodes from three arrays surrounding the sensing region in an outdoor, low-multipath environment. This experiment is more representative of a real-world deployment with multiple anchors. Figure 9 illustrates our setup for the two experiments.

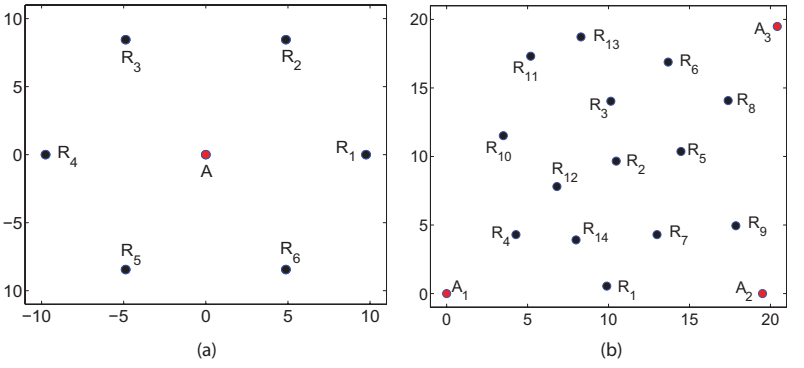


Fig. 9. Experimental setup. (a) Experiment 1. Bearing accuracy of one array. Six receiver nodes ($R_1 \ldots R_6$) are placed 10 meters from array (A), with angular separation of 60°. (b) Experiment 2. Three arrays ($A_1 \ldots A_3$) surround the 20 x 20 m sensing region containing 14 receiver nodes ($R_1 \ldots R_{14}$).

For Experiment 1, we perform 50 bearing estimates for each node surrounding the array. The average bearing errors are displayed in Figure 10a. For Experiment 2, we perform approximately 35 bearing estimates from each anchor to all target nodes, resulting in a total of 105 estimates per target and 1470 estimates total. Figure 10a shows the average error for each bearing from Experiment 1, and the distribution of bearing estimate errors from Experiment 2 are shown in Figure 10b. The average bearing estimation error is 3.2° overall, with a 6.4° accuracy at the 90th percentile. The errors from both experiments are consistent with our bearing error analysis in Section 4.

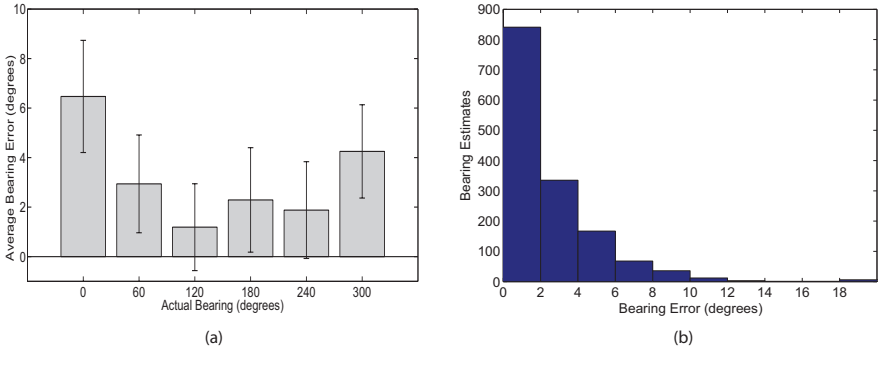


Fig. 10. Experimental results. (a) Experiment 1 average bearing error with respect to array orientation (sample size of 50). (b) Experiment 2 bearing error distribution (sample size of 1470).

In addition, we evaluate the latency of this method. Because we would like to use this system for mobile sensors in addition to stationary nodes, the array must perform its RIMs as fast as possible so that the sensor has not had a chance to significantly change its position. In order to keep the latency to a minimum, we perform an analysis of the different component execution times. Table 1 lists the execution deadlines of the RIM tasks. These deadlines are enforced via software interrupts and were chosen to give each task enough time to complete, assuming a reasonable amount of jitter.

Table 1. Latency of bearing estimation tasks

Task	Latency (ms)
Clock synchronization	162
Acquire and calibrate radio	6.48
Transmit / Receive	63.2
Restore radio driver	49.91
Report phase	250
Bearing estimation	5

The array sends one synchronization message and performs two RIMs, for a total time of 401 ms. An additional 255 ms is required for communication and bearing estimation. Because the target nodes are receivers, no additional latency is incurred by introducing more targets to the sensing region.

7 Conclusion

In this paper, we present a method for rapid distributed bearing estimation in WSNs. The anchor array in our system consists of three nodes, two of which transmit at frequencies that interfere to create a low-frequency beat signal. The phase of this signal is measured by the third node in the array, as well as by multiple target nodes at unknown positions. The phase difference defines a hyperbola, and bearing can be approximated by calculating the angle of the asymptote. Our experimental results show that this technique has an average bearing estimation accuracy of 3.2°, and measurements can be taken in approximately 0.5 sec.

Our system is designed to overcome several challenges in WSN AOA determination. The array prototype is easily constructed by fixing three motes together with antennas at orthogonal angles. It is comprised entirely of COTS sensor nodes, and no additional hardware is required because RIM-based ranging only requires use of the radio. Unlike other radio interferometric techniques, our system avoids the modulo 2π ambiguity, and therefore the need to perform RIMs on multiple channels, by separating the two transmitting antennas less than half the wavelength of the carrier frequency. Similarly, by constraining the location of one of the RIM receivers to the array, it becomes possible to approximate the bearing of the other receiver without prolonged computation or having to rely

on a base station for processing. Our experimental results demonstrate that the accuracy of our prototype implementation is on par with other state-of-the-art AOA techniques.

It is worthwhile noting that this system is designed for eventual use with mobile sensors. Mobility demands rapid localization so that the position estimate is still valid by the time it is computed. Up until now, radio interferometric ranging techniques have been unable to achieve periodic distributed localization at rates fast enough for mobile devices, even slow-moving ones. With this system, we are able to estimate target bearing rapidly enough to support mobile entity localization and navigation. Although at this stage we have not performed tracking or navigation using mobile nodes, we plan on doing so in the near future.

Acknowledgements. This work was supported by ARO MURI grant W911NF-06-1-0076, NSF grant CNS-0721604, and NSF CAREER award CNS-0347440. The authors would also like to thank Peter Volgyesi, Metropolitan Nashville Parks and Recreation, and Edwin Warner Park.

References

1. Niculescu, D., Nath, B.: Ad hoc positioning system (APS) using AOA. In: Proc. of INFOCOM (2003)
2. Esteves, J., Carvalho, A., Couto, C.: Generalized geometric triangulation algorithm for mobile robot absolute self-localization. In: Proc. of ISIE (2003)
3. Biswas, P., Aghajan, H., Ye, Y.: Integration of angle of arrival information for multimodal sensor network localization using semidefinite programming. In: 39th Asilomar Conference on Signals, Systems and Computers (2005)
4. Ash, J.N., Potter, L.C.: Robust system multiangulation using subspace methods. In: Proc. of IPSN (2007)
5. Rong, P., Sichitiu, M.: Angle of arrival localization for wireless sensor networks. In: Proc. of SECON (2006)
6. Nasipuri, A., el Najjar, R.: Experimental evaluation of an angle based indoor localization system. In: 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (2006)
7. Chang, H.l., Tian, J.B., Lai, T.T., Chu, H.H., Huang, P.: Spinning beacons for precise indoor localization. In: Proc. ACM SenSys (2008)
8. Römer, K.: The lighthouse location system for smart dust. In: Proc. of MobiSys (2003)
9. Ash, J.N., Potter, L.C.: Sensor network localization via received signal strength measurements with directional antennas. In: Proceedings of the Allerton Conference on Communication, Control, Computing (2004)
10. Friedman, J., Charbiwala, Z., Schmid, T., Cho, Y., Srivastava, M.: Angle-of-arrival assisted radio interferometry (ARI) target localization. In: Proc. of MILCOM (2008)
11. Maróti, M., Kusý, B., Balogh, G., Völgyesi, P., Nádas, A., Molnár, K., Dóra, S., Lédeczi, A.: Radio interferometric geolocation. In: Proc. of ACM SenSys (2005)
12. Capon, J.: High-resolution frequency-wavenumber spectrum analysis. Proc. of the IEEE 57(8) (1969)
13. Schmidt, R.: Multiple emitter location and signal parameter estimation. IEEE Transactions on Antennas and Propagation 34(3) (1986)

14. Roy, R., Paulraj, A., Kailath, T.: Esprit–a subspace rotation approach to estimation of parameters of cisoids in noise. IEEE Transactions on Acoustics, Speech and Signal Processing 34 (1986)
15. Kumaresan, R., Tufts, D.: Estimating the angles of arrival of multiple plane waves. IEEE Transactions on Aerospace and Electronic Systems AES-19(1) (1983)
16. Kushwaha, M., Amundson, I., Volgyesi, P., Ahammad, P., Simon, G., Koutsoukos, X., Ledeczi, A., Sastry, S.: Multi-modal target tracking using heterogeneous sensor networks. In: Proc. of ICCCN (2008)
17. Priyantha, N.B., Miu, A.K.L., Balakrishnan, H., Teller, S.: The cricket compass for context-aware mobile applications. In: Proc. of MobiCom (2001)
18. Altun, K., Koku, A.: Evaluation of egocentric navigation methods. In: IEEE International Workshop on Robot and Human Interactive Communication (2005)
19. Bekris, K.E., Argyros, A.A., Kavraki, L.E.: Angle-based methods for mobile robot navigation: Reaching the entire plane. In: Proc. of ICRA (2004)
20. Kendig, K.: Conics. Mathematical Association of America (1938)
21. Dutta, P., Grimmer, M., Arora, A., Bibyk, S., Culler, D.: Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In: Proc. of IPSN/SPOTS (2005)
22. Carr, J.: Practical Antenna Handbook, 4th edn. McGraw Hill, New York (2001)
23. Kusý, B., Dutta, P., Levis, P., Maróti, M., Lédeczi, A., Culler, D.: Elapsed time on arrival: a simple and versatile primitive for canonical time synchronization services. International Journal of Ad Hoc and Ubiquitous Computing 2 (2006)

Phoenix: An Epidemic Approach to Time Reconstruction

Jayant Gupchup[1], Douglas Carlson[1], Răzvan Musăloiu-E.[1], Alex Szalay[2], and Andreas Terzis[1]

[1] Computer Science Department
Johns Hopkins University
{gupchup,carlsondc,razvanm,terzis}@jhu.edu
[2] Physics and Astronomy Department
Johns Hopkins University
szalay@jhu.edu

Abstract. Harsh deployment environments and uncertain run-time conditions create numerous challenges for postmortem time reconstruction methods. For example, motes often reboot and thus lose their clock state, considering that the majority of mote platforms lack a real-time clock. While existing time reconstruction methods for long-term data gathering networks rely on a persistent basestation for assigning global timestamps to measurements, the basestation may be unavailable due to hardware and software faults. We present *Phoenix*, a novel offline algorithm for reconstructing global timestamps that is robust to frequent mote reboots and does not require a persistent global time source. This independence sets Phoenix apart from the majority of time reconstruction algorithms which assume that such a source is always available. Motes in Phoenix exchange their time-related state with their neighbors, establishing a chain of transitive temporal relationships to one or more motes with references to the global time. These relationships allow Phoenix to reconstruct the measurement timeline for each mote. Results from simulations and a deployment indicate that Phoenix can achieve timing accuracy up to 6 ppm for 99% of the collected measurements. Phoenix is able to maintain this performance for periods that last for months without a persistent global time source. To achieve this level of performance for the targeted environmental monitoring application, Phoenix requires an additional space overhead of 4% and an additional duty cycle of 0.2%.

1 Introduction

Wireless sensor networks have been used recently to understand spatiotemporal phenomena in environmental studies [12,22]. The data these networks collect are scientifically useful only if the collected measurements have corresponding, accurate global timestamps. The desired level of accuracy in this context is in the order of milliseconds to seconds. In order to reduce complexity of the code running on the mote, it is more efficient to record sensor measurements using the mote's local time frame and perform a postmortem reconstruction to translate them to global time.

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 17–32, 2010.

Each mote's clock (referred to as local clock henceforth) monotonically increases and resets to zero upon reboot. A naive postmortem time reconstruction scheme collects $\langle local, global \rangle$ pairs during a mote's lifetime, using a global clock source (typically, an NTP-synchronized PC). These pairs (also referred to as "anchor points") are then used to translate the collected measurements to the global time frame by estimating the motes' clock skew and offset. We note that this methodology is unnecessary for architectures such as Fleck, which host a battery-backed on-board real-time clock (RTC) [4]. However, many commonly-used platforms such as Telos, Mica2, MicaZ, and IRIS (among others) lack an on-board RTC.

In the absence of reboots, naive time reconstruction strategies perform well. However, in practice, motes reboot due to low battery power, high moisture, and software defects. Even worse, when motes experience these problems, they may remain completely inactive for non-deterministic periods of time. Measurements collected during periods which lack $\langle local, global \rangle$ anchors (due to rapid reboots and/or basestation absence) are difficult or impossible to accurately reconstruct. Such situations are not uncommon based on our deployment experiences and those reported by others [23].

In this work, we devise a novel time reconstruction strategy, *Phoenix*, that is robust to random mote reboots and intermittent connection to the global clock source. Each mote periodically listens for its neighbors to broadcast their local clock values. These $\langle local, neighbor \rangle$ anchors are stored on the mote's flash. The system assumes that one or more motes can periodically obtain global time references, and they store these $\langle local, global \rangle$ anchors in their flash. When the basestation collects the data from these motes, an offline procedure converts the measurements timestamped using the motes' local clocks to the global time by using the transitive relationships between the local clocks and global time.

The offline nature of Phoenix has two advantages: **(a)** it reduces the complexity of the software running on the mote, and **(b)** it avoids the overhead associated with executing a continuous synchronization protocol. We demonstrate that Phoenix can reconstruct global timestamps accurately (within seconds) and achieve low ($< 1\%$) data losses in the presence of random mote reboots even when months pass without access to a global clock source.

2 Motivation

We claim that the problem of rebooting motes is a practical aspect of real deployments that has a high impact on environmental monitoring applications. We also quantify the frequency and impact of reboots in a long-term deployment. We begin by understanding why mote reboots complicate postmortem time reconstruction.

2.1 Postmortem Timestamp Reconstruction

The relationship between a mote's local clock, LTS, and the global clock, GTS, can be modeled with a simple linear relation: $GTS = \alpha \times LTS + \beta$, where α

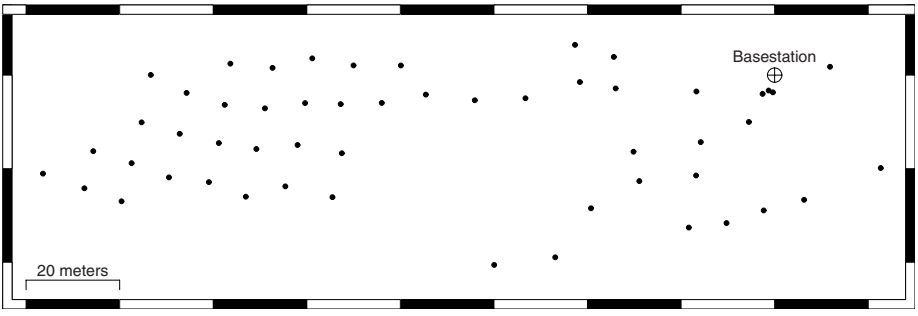


Fig. 1. The 53-mote "Cub Hill" topology, located in an urban forest northeast of Baltimore, Maryland

represents the mote's skew and β represents the intercept (global time when the mote reset its clock) [19]. This conversion from the local clock to global clock holds as long as the mote's local clock monotonically increases at a constant rate. We refer to this monotonically increasing period as a *segment*. When a mote reboots and starts a new segment, one needs to re-estimate the fit parameters. If a mote reboots multiple times while it is out of contact with the global clock source, estimating β for these segments is difficult. While data-driven treatments have proven useful for recovering temporal integrity, they cannot replace accurate timestamping solutions [9,10]. Instead, time reconstruction techniques need to be robust to mote reboots and not require a persistent global time source.

2.2 Case Studies

We present two cases which illustrate the deployment problems that Phoenix intends to address. The first is an account of lessons learned from a year-long deployment of 53 motes. The second is a result of recent advances in solar-powered sensor networks.

Software Reboots. We present "Cub Hill", an urban forest deployment of 53 motes that has been active since July 2008 (Figure 1). Sensing motes collect measurements every 10 minutes to study the impact of land use on soil conditions. The basestation uses the Koala protocol to collect data from these motes every six hours [15]. We use TelosB motes driven by 19 Ah, 3.6 V batteries.

We noticed that motes with low battery levels and/or high internal moisture levels suffered from periodic reboots. As an example, Figure 2 shows the battery voltage of a mote that rebooted thrice in one month. Despite their instability, many of these motes were able to continue collecting measurements for extended periods of time.

Following a major network expansion, a software fault appeared which caused nodes to "freeze". Unable to reproduce this behavior in a controlled environment, we employed the MSP430's Watchdog Timer to reboot motes that enter this state [21]. While this prevented motes from completely failing, it also shortened

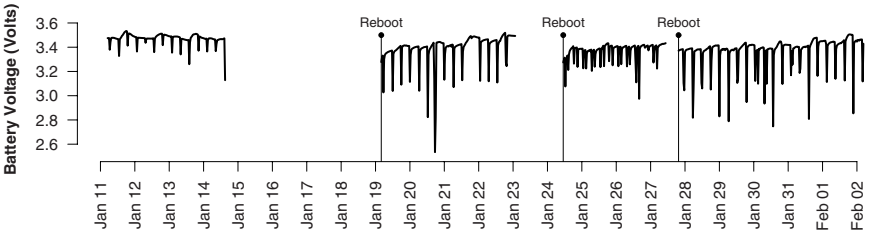


Fig. 2. An example of a mote rebooting due to low battery voltage (no watchdog timer in use). The sharp downward spikes correspond to gateway downloads (every six hours). Gaps in the series are periods where the mote was completely inoperative.

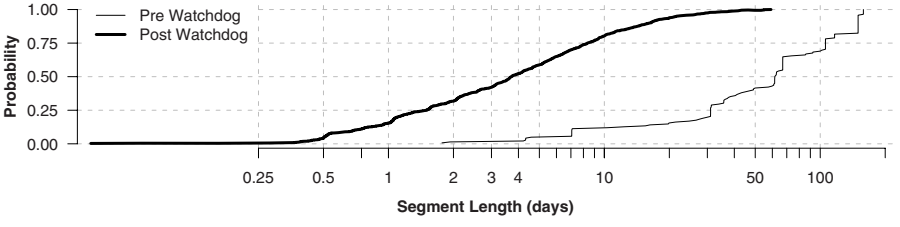


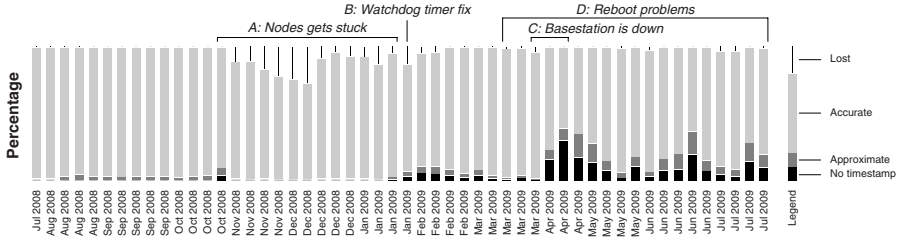
Fig. 3. The distribution of the segment lengths before and after adding the watchdog timer to the mote software

the median length of the period between reboots from more than 50 days to only four days, as Figure 3 shows.

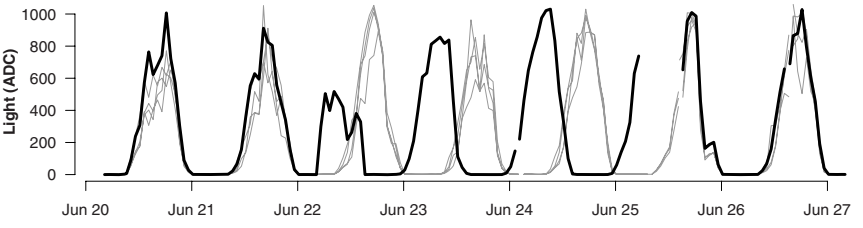
Solar Powered Sensor Networks. A number of research groups have demonstrated the use of solar energy as a means of powering environmental monitoring sensor networks [11,20]. In such architectures, a mote can run out of power during cloudy days or at night. Motes naturally reboot in such architectures, and data losses are unavoidable due to the lack of energy. It is unclear how one can achieve temporal reliability without a persistent basestation or an on-board RTC. To the best of our knowledge, no one has addressed the issue of temporal integrity in solar-powered sensor networks. Yang et al. employ a model in which data collection happens without a persistent basestation [24]. The data upload takes place infrequently and opportunistically. Hard-to-predict reboot behavior is common to these systems. Furthermore, we note that even though there is very little information about the rate of reboots in such architectures, it is clear that such systems are susceptible to inaccurate timestamp assignments.

2.3 Impact

We evaluate the impact of mote reboots on the Cub Hill deployment using our existing time reconstruction methodology.



(a) The fraction of measurements that were assigned timestamps.



(b) An example of the impact of estimating β incorrectly when using approximate methods. Data from one of the motes (represented with the dark line) that rebooted multiple times between Jun. 22 and Jun. 25. During this period, the mote was out of sync with the rest (shown in gray) due to inaccurate β estimates

Fig. 4. Impact of time reconstruction methodology using the RGTR algorithm

The basestation records an anchor point each time it downloads data from a mote. Motes that are poorly connected to the basestation may remain out of contact for several download rounds before connectivity improves and they can transfer their data. When motes reboot at a rate faster than the frequency with which the basestation contacts them, there exist periods which lack enough information to accurately reconstruct their measurement timestamps.

Upon acquiring the anchor points, the measurements are converted from their local clock to the global clock at the basestation. We employ our previously proposed algorithm, Robust Global Timestamp Reconstruction algorithm (referred to as RGTR), for this purpose [9]. We note that in order to estimate the fit parameters (α, β) for the segments, RGTR requires at least two anchor points. Depending on the accuracy requirements, one can assume that the skew (α) is stable per mote for small segments. Using this assumption, at least one anchor point is needed to estimate the β for any given segment, provided that α has been estimated accurately for the mote.

Figure 4(a) demonstrates the impact of mote reboots on time reconstruction for the Cub Hill deployment. During period A, motes were prone to freezing (and thus stopped sampling), leading to a decrease in the total data collected. At point B, the addition of the watchdog timer caused the total data collected to return to its previous level. However, due to the increased frequency of reboots, a larger portion of the samples could not be assigned a global timestamp (exacerbated by the absence of the base station during period C).

For segments where no anchor points were collected, we assumed that node reboots are instantaneous. However, this assumption does not always hold (see Figure 2) and leads to a small fraction of misaligned measurements. Figure 4(b) presents an example of this misalignment. One node (shown in bold) rebooted multiple times and could not reach the basestation during its active periods. The assumption of instantaneous reboots led to inaccurate β estimates.

3 Solution

Phoenix is a postmortem time reconstruction algorithm for motes operating without in-network time synchronization. It consists of two stages.

3.1 In-Network Anchor Collection

Each mote operates solely with respect to its own local clock. A new segment (uniquely identified by $\langle moteid, reboot\ count \rangle$) begins whenever a mote reboots: each segment starts at a different time and may run at a different rate. Our architecture assumes that there is at least one mote in the network that can periodically obtain references from an accurate global time source. This is done to establish the global reference points needed by Phoenix. This source may be absent for long periods of time (see Section 4). The global time source can be any reliable source (a mote equipped with a GPS receiver, NTP-synced basestation, etc). Without loss of generality, we assume that the network contains a mote connected to GPS device and a basestation that collects data infrequently[1].

All motes (including the GPS-connected mote) broadcast their local clock and reboot-count values every T_{beacon} seconds. Each receiving mote stores this information (along with its own local clock and reboot counter) in flash to form anchor records. The format of these records is $\langle moteid_r, rc_r, lc_r, moteid_s, rc_s, lc_s \rangle$; where rc, lc, r, and s refer to the reboot counter, local clock, receiver and sender respectively. Periodically, motes turn on their radios and listen for broadcasts in order to anchor their time frame to those of their neighbors. Each mote tries to collect this information from its neighbors after every reboot and after every T_{wakeup} seconds ($\gg T_{beacon}$). The intuition behind selecting this strategy is as follows. The reboot time determines the β parameter. The earliest opportunity to extract this information is immediately after a reboot. To get a good estimate of the skew, one would like to collect multiple anchors that are well distributed in time. Thus, T_{wakeup} is a parameter that governs how far to spread out anchor collections. In the case of a GPS mote, the $moteid_r, rc_r$ and $moteid_s, rc_s$ are identical, and lc_r, lc_s represent the local and global time respectively.

The basestation periodically downloads these anchors along with the measurements. This information is then used to assign global timestamps to the collected measurements using Algorithm 1. If the rate of reboots is known, the anchor collection frequency can be fixed conservatively to collect enough anchors

[1] Note that the basestation collects data *only* and it does not provide a time source, unless specified otherwise.

Algorithm 1. Phoenix

Definitions:
a, b : alpha and beta for local-local fits;
P : parent segment; Π : Ancestor segments

procedure PHOENIX(AP)
  for each (i, j) in KEYS(AP) **do** ▷ All unique segment pairs in AP
    $LF_{a,b,\chi,df}(i, j) \leftarrow$ LSE($AP(i, j)$) ▷ Compute the local-local fits
  for each $s \in S$ **do** ▷ Set of all unique segments
    $GF_{\alpha,\beta,P,\Pi,\chi,df}(s) \leftarrow (\emptyset, \emptyset, \emptyset, s, \chi_{MAX}, \emptyset)$ ▷ Initialize global fits
  for each $g \in G$ **do** ▷ All segments anchored to GTS
    INITGTSNODES(g, LF, GF)
    ENQUEUE(Q, g) ▷ Add all the GTS nodes to the queue
  while NOTEMPTY(Q) **do**
    $q \leftarrow$ DEQUEUE(Q)
    $C \leftarrow$ NEIGHBORANCHORS(q)
    for each $c \in C$ **do**
      $T_{\alpha,\beta,P,\Pi,\chi,df}(c) \leftarrow$GLOBALFIT($c, q, GF, LF$)
      if (UPDATEFIT(c, T, GF)) **then** ▷ Check for a better fit
        ENQUEUE(C)
  return GF

procedure INITGTSNODES(g, LF, GF)
  $GF(g) \leftarrow (LF_a(g, g'), LF_b(g, g'), \emptyset, g, LF_\chi(g, g'), LF_{df}(g, g'))$ ▷ g' is GTS, g is LTS

procedure GLOBALFIT(c, q, GF, LF)
  if $q > c$ **then** ▷ Smaller segment is the independent variable
    $\alpha_{new} \leftarrow GF_\alpha(q) * LF_a(q, c)$
    $\beta_{new} \leftarrow GF_\alpha(q) * LF_b(q, c) + GF_\beta(q)$
  else
    $\alpha_{new} \leftarrow GF_\alpha(q)/LF_a(q, c)$
    $\beta_{new} \leftarrow GF_\alpha(q) - \alpha_{new} * LF_b(q, c)$
  $\chi \leftarrow \frac{GF_{df}(q)*GF_\chi(q)+LF_{df}(q,c)*LF_\chi(q,c)}{GF_{df}(q)+LF_{df}(q,c)}$ ▷ Compute the weighted GOF metric.
  $df \leftarrow GF_{df}(q) + LF_{df}(q, c)$
  return $(\alpha_{new}, \beta_{new}, q, \{c \cup GF_\Pi(q)\}, \chi, df)$ ▷ Update parent/ancestors

procedure UPDATEFIT(c, T, GF)
  if $c \in T_\Pi(c)$ **then** ▷ Check for cycles
    return false
  if $T_\chi(c) < GF_\chi(c)$ **then**
    $GF_{\alpha,\beta,P,\Pi,\chi,df}(c) \leftarrow T_{\alpha,\beta,P,\Pi,\chi,df}(c)$
    return true
  else
    return false

between reboots. One could also employ an adaptive strategy by collecting more anchors when the segment is small and reverting to a larger T_{wakeup} when an adequate number of anchors have been collected. It is advantageous for a mote to attempt to collect anchors from a small set of neighbors (to minimize storage), but this requires a mote to have some way of identifying the most useful segments for anchoring (see Section 4).

3.2 Offline Timestamp Reconstruction

The Phoenix algorithm is intuitively simple. We will outline it in text and draw attention to a few important details. For a more complete treatment, please refer to the pseudocode in Algorithm 1. Phoenix accepts as input the collection of all anchor points AP (both $\langle local, neighbor \rangle$ and $\langle local, global \rangle$). It then employs

a least-square linear regression to extract the relationships between the local clocks of the segments that have anchored to each other (LF, for Local Fit). In addition to $LF_a(i,j)$ (slope), $LF_b(i,j)$ (intercept), Phoenix also obtains a goodness-of-fit (GOF) metric, $LF_\chi(i,j)$ (unbiased estimate of the variance of the residuals) and LF_{df} (degrees of freedom). For segments which have global references, Phoenix stores this as GF (for Global Fit).

The algorithm then initializes a queue with all of the segments which have direct anchors to the global clock. It dequeues the first element q and examines each segment c that has anchored to it. Phoenix uses the transitive relationship between $GF(q)$ and $LF(q,c)$ to produce a global fit $T(c)$ which associates segment c to the global clock through segment q. If $T_\chi(c)$ is lower than the previous value for $GF_\chi(c)$ (and using q would not create a cycle in the path used to reach the global clock), the algorithm replaces $GF(c)$ with $T(c)$, and places c in the queue. When the queue is empty, no segments have "routes" to the global clock which have a better goodness-of-fit than the ones which have been previously established. At this point, the algorithm terminates.

The selection of paths from an arbitrary segment to a segment with global time references can be thought of as a shortest-path problem (each segment represents a vertex and the fit between the two segments is an edge). The GOF metric represents the edge weight. The running time complexity of the implementation of Phoenix was validated experimentally by varying the deployment lifetime (thereby varying number of segments). The runtime was found to increase slower than the square of the number of segments.

4 Evaluation

We evaluate the effect of varying several key parameters in Phoenix using both simulated and real datasets. We begin by describing our simulator.

4.1 Simulator

Our goal is to minimize the data loss in long-term deployments. Hence, we fix the simulation period to be one year. We also assume that the basestation is not persistently present and does not provide a time source to the network. The network contains one global clock source (a GPS mote) that is susceptible to failures. The main components of the simulator are described below. The default values for the simulator are based on empirical data obtained from the one year long Cub Hill deployment.

Clock Skew: The clock skew for each segment is drawn from a uniformly distributed random variable between 40 ppm and 70 ppm. Burri et al. report this value to be between 30 and 50 ppm at room temperature[2] [1].

Segment Model: We use the non-parametric segment-length model based on the Cub Hill deployment after the watchdog timer fix (Figure 3). Additionally,

[2] We ignore the well-studied temperature effects on the quartz crystal. For a more complete treatment on the temperature dependence, refer to [14,17].

after a reboot, we allowed the mote to stay inactive for a period that is randomly drawn between zero and four hours with a probability given by $p_{down} = 0.2$. The GPS mote's behavior follows the same model.

Communication Model: The total end-to-end communication delay for receiving anchor packets is drawn uniformly between 5 and 15 milliseconds. This time includes the interrupt handling, transmission, reception and propagation delays. To model the packet reception rate (PRR), we use the log-distance path loss model as described in [18,25] with parameters: $(P_r(d_0), \eta, \sigma, d_0) = (-59.28, 2.04, 6.28, 2.0m)$.

Topology: The Cub Hill topology was used as the basis for all simulations.

Event Frequencies: Motes recorded a 26-byte sample every 10 minutes. They beacon their local clock values with an interval of T_{beacon}. They stay up after every reboot and periodically after an interval of T_{wakeup} to collect these broadcasts. While up, they keep their radios on for a maximum of T_{listen}. The GPS mote collects $\langle local, global \rangle$ anchors with a rate of T_{sync}. By default, T_{beacon}, T_{wakeup}, T_{listen} and T_{sync} were set to 30 s, 6 h, 30 s and 6 h respectively.

Maximum Anchorable Segments: To minimize the space overhead in storing anchors, we limit the number of segments that can be used for anchoring purposes. At any given time, a mote can only store anchors for up to $NUMSEG$ segments. The default $NUMSEG$ value is set to four. Motes stop listening early once they collect $NUMSEG$ anchors in a single interval.

Eviction Policy: Since segments end and links between motes change over time, obsolete or rarely-heard segments need to be evicted from the set of $NUMSEG$ segments for which a mote listens. The timeout for evicting stale entries is set to $3 \times T_{wakeup}$. We evaluated three different strategies for selecting replacements for evicted segments. First-come, first-served (FCFS) accepts the first segment that is heard when a vacancy exists. RAND keeps track of the previous segments that were heard and selects a new segment to anchor with at random. Longest local clock (LLC) keeps track of the local clock values of the segments that are heard and selects the segment that has the highest local clock. FCFS was chosen as the default.

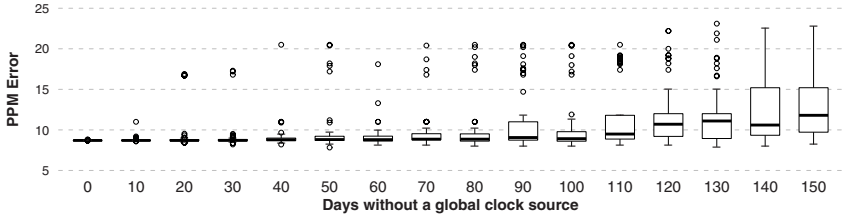
4.2 Evaluation Metrics

Data loss (DL): The fraction of data that cannot be assigned any timestamps, expressed as a percentage.

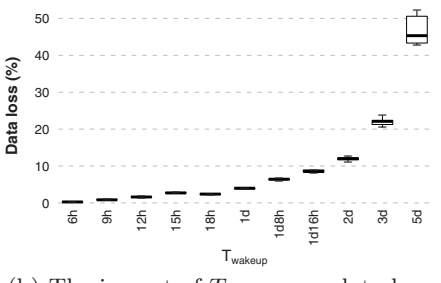
PPM Error: The average error (in parts per million) for the assigned timestamps. PPM error is $\frac{|t'-t|}{t_\delta} \times 10^6$, where t is the true timestamp of the measurement, t' is the assigned timestamp, and t_δ denotes the elapsed time since the start of the segment in terms of the real clock.

Space overhead: The fraction of space that is used for storing anchors relative to the total space used, expressed as a percentage.

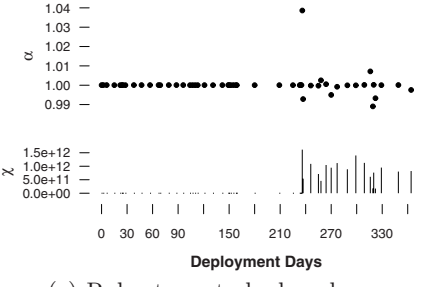
Duty cycle: The fraction of time the radio was kept on for anchor collection and beaconing, expressed as a percentage.



(a) The effect of a missing global clock source on accuracy.



(b) The impact of T_{wakeup} on data loss.



(c) Robustness to bad anchors.

Fig. 5. Evaluation of Phoenix in simulation. In (c), faults were injected to GPS anchors after day 237. Figure shows the α and χ values for the GPS mote for the entire period.

4.3 Simulation Experiments

Dependence on Global Clock Source: We studied the effect of the global clock's absence on data loss. We assume that the network contains one GPS mote that serves as the global clock source and it is inoperative for a specified amount of time. In order to avoid bias, we randomly selected the starting point of this period and varied the GPS down time from 0 to 150 days in steps of 10. Figure 5(a) shows the effect on the reconstruction using 60 independent runs. The accuracy decreases as the number of days without GPS increases, but we note that this decrease is tolerable for our target applications. The data loss stayed relatively stable at 0.21%, even when the global clock source is absent for as long as 5 months. We note that in a densely connected network, the number of paths between any two segments is combinatorial, and hence, the probability of finding a usable path is very high[3]. The variance of the error increased with the length of the gateway's absence.

Dependence on Wake-up Interval: Figures 5(b) show the effect of varying wake-up rate on data loss. As expected, data loss increases as the rate of anchor collection decreases. This curve is strongly related to the segment model: if

[3] One can estimate the probability for finding a usable path using Warshall's algorithm [5]. The input to this algorithm would be a connectivity matrix where the entries represent the anchoring probabilities of the neighbor segments.

collections are less frequent than reboots, many segments will fail to collect enough anchors to be reconstructed.

Robustness: We studied the effect of faulty global clock references on time reconstruction. Noise from a normal distribution ($\mu = 60$ min., $\sigma = 10$ min.) was added to the global references for a period of 128 days. Figure 5(c) shows the alpha and χ values for the GPS mote during the entire simulation period. One can also notice the correlation between high χ values and α values that deviate from 1.0 in Figure 5(c). These faults did not change the data loss rate. The faults increased the PPM error from 4.03 to 16.5. Although these faults decreased accuracy, this decrease is extremely small in comparison to the magnitude of the injected errors and within the targeted accuracy requirements. Phoenix extracted paths which were least affected by these faults by using the χ metric.

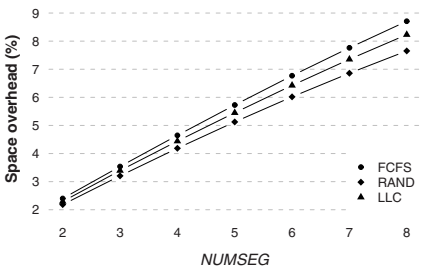
Effect of eviction and $NUMSEG$: We studied the effect of $NUMSEG$ on space, duty cycle, and data loss. The space overhead increases linearly with $NUMSEG$ (Figure 6(a)). The impact on duty cycle[4] was quite low (Figure 6(b)). A constant duty cycle penalty of 0.075% is incurred due to the beaconing messages sent every 30 s [16]. At low values of $NUMSEG$, motes are able to switch off their radios early (once they have heard announcements from segments they have anchored with), while at higher values, they need to stay on for the entire T_{listen} period. Increasing $NUMSEG$ decreases data loss, because motes have a better chance of collecting good segments to anchor with. We found that the FCFS eviction policy outperforms LLC and RAND. We found no significant differences in the PPM error results as we vary $NUMSEG$, and hence, we do not report those results here.

Neighbor Density: In this experiment, we removed links from the Cub Hill topology until we obtained the desired neighbor density. At every step, we ensured that the network was fully connected. We did not find any significant impact on performance as the average number of neighbors was decreased. In this experiment, the radios were kept on for the entire T_{listen} period, and no eviction policy was employed. This was done to compare the performance at each density level at the same duty cycle. Figure 6(d) presents our findings.

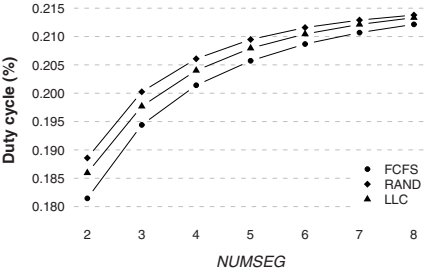
4.4 Deployment

We deployed a network (referred to as the "Olin" network) of 19 motes arranged in a grid topology in an urban forest near the Johns Hopkins University campus in Baltimore, MD. Anchors were collected for the entire period of 21 days using the methodology described in Section 3.1. The basestation collected data from these motes once every four hours and the NTP-corrected clock of the basestation was used as a reliable global clock source. The motes rebooted every 5.7 days on

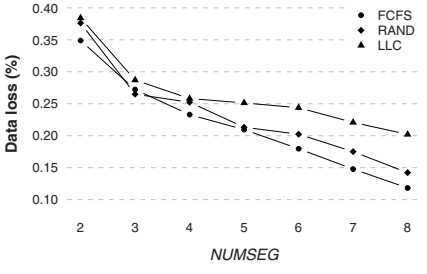
[4] Note that the duty cycle that we are referring to does not consider the communication costs during data downloads. Reducing the storage requirements would reduce the communication costs when the basestation collects data.



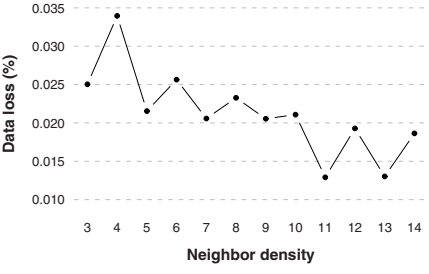
(a) Space overhead in storing anchors as a function of $NUMSEG$.



(b) Duty cycle as a function of $NUMSEG$.



(c) Data loss as a function of $NUMSEG$.



(d) Effect of varying node density on data loss with no eviction policy.

Fig. 6. Effect of $NUMSEG$ on different eviction policies

average, resulting in a total of 62 segments. The maximum segment length was 19 days and the minimum was two hours.

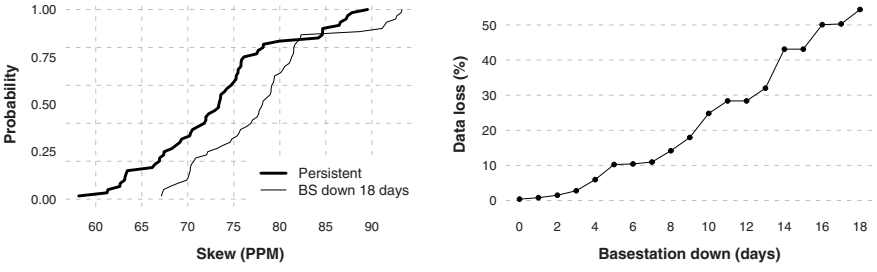
Perceived Ground Truth: It is very difficult to establish absolute ground truth in field experiments. Instead, we establish a synthetic ground truth by reconstructing timestamps using all the global anchors obtained from the basestation[5]. We record the α and β values for each segment and use these values as ground truth. Because we downloaded data every four hours we obtained enough global anchors from the motes to be confident with the derived ground truth estimates.

Emulating GPS node and Basestation Failure: In order to emulate a GPS mote, we selected a single mote (referred to as G-mote) that was one hop away from the basestation. We used the G-mote's global anchors obtained from the basestation as though they were taken using a GPS device. We ignored all other global anchors obtained from other motes. Furthermore, to emulate the absence of the basestation for N days, we discarded all the anchors taken by the G-mote during that N-day long period. We tested for values of N from one to eighteen.

[5] Note that every time a mote contacts the basestation, we obtain a global anchor for that mote.

Table 1. Phoenix accuracy using the Olin dataset as a function of the number of days that the basestation was unavailable

Error\Days	2	4	6	8	10	12	14	16	18
α_{med} (ppm)	1.73	1.73	1.85	1.70	1.96	2.20	4.36	5.47	5.93
α_{std} (ppm)	3.41	3.40	3.40	3.39	3.30	3.26	3.17	3.00	3.00
β_{med} (s)	0.88	0.88	0.91	0.94	1.16	1.55	4.52	6.02	6.44
β_{std} (s)	0.58	0.57	0.58	0.57	0.65	0.91	2.43	3.11	3.45



(a) The CDF of α estimates on the Olin deployment

(b) Data loss using RGTR. Data loss from Phoenix was $< 0.06\%$.

Fig. 7. The stability of the α estimates using Phoenix and the data loss using RGTR in comparison to Phoenix

Phoenix Accuracy: After simulating the basestation failure, we reconstruct the timestamps by applying Phoenix using only the $\langle local, neighbor \rangle$ anchors, and global anchors available from the G-mote. This provides us with another set of α and β estimates for each of the segments. We compare these estimates with the ground truth estimates (pair-wise comparison). In order to provide a deeper insight, we decompose the average PPM error metric into its constituent components - α and β errors. Furthermore, we report the median and standard deviation of these α and β errors. Table 1 reports the results of these experiments. We found that the median α error stayed as low as 5.9 ppm, while the median β error stayed as low as 6.4 s for N =18. In general, α_{med}, β_{med} and β_{std} increased as N increased and α_{std} stayed relatively consistent for different values of N. The stability of the α estimates using Phoenix with $N = 0$ and $N = 18$ is shown in Figure 7(a). The CDF shows that median skew was found to be around 75 ppm and the two curves track each other closely.

Data Loss: The data loss using Phoenix was found to be as low as 0.055% when N was 18 days. In comparison, we found that there was significant data loss when the timestamps were reconstructed using RGTR. Figure 7(b) shows the data losses for different values of N. The figure does not report the Phoenix data loss as we found it to be 0.055% irrespective of N. This demonstrates that Phoenix is able to reconstruct more than 99% of the data even when motes

reboot frequently and the basestation is unavailable for days. We note that in comparison to Phoenix, RGTR does not incur any additional storage and duty cycle overheads as anchors are recorded at the basestation directly as part of the data downloads.

5 Related Work

Assignment of timestamps in sensor networks falls under two broad categories. Strict clock synchronization aims at ensuring that all the mote clocks are synchronized to the same clock source. Flooding Time Synchronization Protocol (FTSP, [13]), Reference Broadcast Synchronization (RBS, [7]), and the Timing-sync Protocol for Sensor Networks [8] are examples of this approach. These systems are typically used in applications such as target tracking and alarm detection which require strong real-time guarantees of reporting events. The second category is known as postmortem time reconstruction and it is mostly used due to its simplicity. While strict synchronization is appropriate for applications where there are specific events of interest that need to be reported, postmortem reconstruction is well-suited for applications where there is a continuous data stream and every measurement requires an accurate timestamp.

Phoenix falls under the second class of methods. The idea of using linear regression to translate local timestamps to global timestamps was first introduced by Werner-Allen et al. in a deployment that was aimed at studying active volcanoes [23]. This work, however, does not consider the impact caused by rebooting motes and basestation failures from a time reconstruction perspective. More recently, researchers have proposed data-driven methods for recovering temporal integrity [9,10]. Lukac et al. use a model for microseism propagation to time-correct the data collected by their seismic sensors. Although data-driven methods have proved useful for recovering temporal integrity, they are not a solution for accurate timestamping.

Routing integrated time synchronization protocol (RITS, [19]) spans these categories. Each mote along the path (to the basestation) transforms the time of the reported event from the preceding mote's time frame, ending with an accurate global timestamp at the basestation. RITS does not consider the problem of mote reboots, and is designed for target tracking applications. The problem of mote reboots have been reported by a number of research groups. Chang et al. report that nodes rebooted every other day due to an unstable power source [2], whereas Dutta et al. employed the watchdog timer to reboot nodes due to software faults [6]. Allen et al. report an average node uptime of 69% [23]. More recently, Chen et al. advocate *Neutron*, a solution that detects system violations and recovers from them without having to reboot the mote [3]. They advocate the notion of preserving "precious" states such as the time synchronization state. Nevertheless, Neutron cannot prevent all mote reboots and therefore Phoenix is still necessary.

6 Conclusions

In this paper we investigate the challenges facing existing postmortem time reconstruction methodologies due to basestation failures, frequent random mote reboots, and the absence of on-board RTC sources. We present our time reconstruction experiences based on a year-long deployment and motivate the need for robust time reconstruction architectures that minimize data losses due to the challenges we experienced.

Phoenix is an offline time reconstruction algorithm that assigns timestamps to measurements collected using each mote's local clock. One or more motes have references to a global time source. All motes broadcast their time-related state and periodically record the broadcasts of their neighbors. If a few mote segments are able to map their local measurements to the global time frame, this information can then be used to assign global timestamps to the measurements collected by their neighbors and so on. This epidemic-like spread of global information makes Phoenix robust to random mote reboots and basestation failures. We found that in practice there are more than enough possible ways to obtain good fits for the vast majority of data segments.

Results obtained from simulated datasets showed that Phoenix is able to timestamp more than 99% of measurements with an accuracy up to 6 ppm in the presence of frequent random mote reboots. It is able to maintain this performance even when there is no global clock information available for months. The duty-cycle and space overheads were found to be as low as 0.2% and 4% respectively. We validated these results using a 21 day-long real deployment and were able to reconstruct timestamps in the order of seconds.

In the future, we will investigate using other metrics for determining edge weights and their impact on the quality of the time reconstruction. Moreover, we will explore adaptive techniques for determining the anchor collection frequency. Finally, we will derive theoretical guarantees on the accuracy of Phoenix, which can be used to allow for fine-grained tradeoffs between reconstruction quality and overhead.

Acknowledgments

We thank Prabal Dutta, Jay Taneja and the anonymous reviewers for their comments that helped us to improve the paper's presentation. This research was supported in part by NSF grants DBI-0754782 and CNS-0720730. Any opinions, finding, conclusions or recommendations expressed in this publication are those of the authors and do not represent the policy or position of the NSF.

References

1. Burri, N., von Rickenbach, P., Wattenhofer, R.: Dozer: ultra-low power data gathering in sensor networks. In: IPSN (2007)
2. Chang, M., Cornou, C., Madsen, K., Bonnet, P.: Lessons from the Hogthrob Deployments. In: WiDeploy (June 2008)

3. Chen, Y., Gnawali, O., Kazandjieva, M., Levis, P., Regehr, J.: Surviving sensor network software faults. In: SIGOPS (October 2009)
4. Commonwealth Scientific and Industrial Research Organisation (CSIRO). 2-year progress report: July 2004 to June 2006 (2004)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. McGraw-Hill Science/Engineering/Math, New York (2001)
6. Dutta, P., Hui, J., Jeong, J., Kim, S., Sharp, C., Taneja, J., Tolle, G., Whitehouse, K., Culler, D.: Trio: Enabling sustainable and scalable outdoor wireless sensor network deployments. In: IEEE SPOTS, pp. 407–415 (2006)
7. Elson, J.E., Girod, L., Estrin, D.: Fine-grained network time synchronization using reference broadcasts. In: OSDI, December 2002, pp. 147–163 (2002)
8. Ganeriwal, S., Kumar, R., Srivastava, M.B.: Timing-sync protocol for sensor networks. In: Proceedings of SensSys, November 2003, pp. 138–149 (2003)
9. Gupchup, J., Musaloiu-Elefteri, R., Szalay, A.S., Terzis, A.: Sundial: Using sunlight to reconstruct global timestamps. In: Roedig, U., Sreenan, C.J. (eds.) EWSN 2009. LNCS, vol. 5432, pp. 183–198. Springer, Heidelberg (2009)
10. Lukac, M., Davis, P., Clayton, R., Estrin, D.: Recovering temporal integrity with data driven time synchronization. In: IPSN, April 2009, pp. 61–72 (2009)
11. Luo, L., Huang, C., Abdelzaher, T., Stankovic, J.: EnviroStore: A cooperative storage system for disconnected operation in sensor networks. In: INFOCOM (2007)
12. Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., Anderson, J.: Wireless sensor networks for habitat monitoring. In: WSNA, pp. 88–97. ACM, New York (2002)
13. Maróti, M., Kusy, B., Simon, G., Lédeczi, A.: The flooding time synchronization protocol. In: SenSys, November 2004, pp. 39–49 (2004)
14. Marrison, W.A.: The evolution of the quartz crystal clock. The Bell System Technical Journal 27 (1948)
15. Musaloiu-E., R., Liang, C.-J.M., Terzis, A.: Koala: Ultra-low power data retrieval in wireless sensor networks. In: IPSN, pp. 421–432 (2008)
16. Musăloiu-E., R., Liang, C.-J.M., Terzis, A.: Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. In: Proceedings of the Seventh International Conference on Information Processing in Sensor Networks (IPSN) (April 2008)
17. Newell, D.E., Bangert, R.H.: Temperature compensation of quartz crystal oscillators. In: 17th Annual Symposium on Frequency Control 1963, pp. 491–507 (1963)
18. Rappaport, T.S.: Wireless Communications: Principles and Practice, 2nd edn. Prentice Hall PTR, Englewood Cliffs (2002)
19. Sallai, J., Kusy, B., Lédeczi, Á., Dutta, P.: On the scalability of routing integrated time synchronization. In: Römer, K., Karl, H., Mattern, F. (eds.) EWSN 2006. LNCS, vol. 3868, pp. 115–131. Springer, Heidelberg (2006)
20. Taneja, J., Jeong, J., Culler, D.: Design, modeling, and capacity planning for micro-solar power sensor networks. In: IPSN 2008, pp. 407–418 (2008)
21. Texas Instruments Incorporated. MSP430 Datasheet
22. Tolle, G., Polastre, J., Szewczyk, R., Turner, N., Tu, K., Buonadonna, P., Burgess, S., Gay, D., Hong, W., Dawson, T., Culler, D.: A Macroscope in the Redwoods. In: SenSys (November 2005)
23. Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., Welsh, M.: Fidelity and Yield in a Volcano Monitoring Sensor Network. In: OSDI (November 2006)
24. Yang, Y., Wang, L., Noh, D.K., Le, H.K., Abdelzaher, T.F.: Solarstore: enhancing data reliability in solar-powered storage-centric sensor networks. In: Mobisys, pp. 333–346. ACM, New York (2009)
25. Zamalloa, M.Z., Krishnamachari, B.: An analysis of unreliability and asymmetry in low-power wireless links. ACM Trans. Sen. Netw. 3(2), 7 (2007)

Trimming the Tree: Tailoring Adaptive Huffman Coding to Wireless Sensor Networks

Andreas Reinhardt[1], Delphine Christin[2], Matthias Hollick[2], Johannes Schmitt[1], Parag S. Mogre[1], and Ralf Steinmetz[1]

[1] Multimedia Communications Lab, Technische Universität Darmstadt
Rundeturmstr. 10, 64283 Darmstadt, Germany
{areinhardt,jschmitt,pmogre,ralf.steinmetz}@kom.tu-darmstadt.de
[2] Secure Mobile Networking Lab, Technische Universität Darmstadt
Mornewegstr. 32, 64293 Darmstadt, Germany
{delphine.christin,matthias.hollick}@seemoo.tu-darmstadt.de

Abstract. Nodes in wireless sensor networks are generally designed to operate on a limited energy budget, and must consciously use the available charge to allow for long lifetimes. As the radio transceiver is the predominant power consumer on current node platforms, the minimization of its activity periods and efficient use of the radio channel are major targets for optimization. Data compression is a viable option to increase the packet information density, resulting in reduced transmission durations and thus allowing for an optimized channel utilization. The computational and memory demands of many current compression algorithms however hamper their applicability on sensor nodes.

In this paper, we present a novel variant of the adaptive Huffman coding algorithm, operating on reduced code table sizes and thus significantly alleviating the resource demands for storing and updating the code table during runtime. An implementation for tmote sky hardware proves its adequacy to the capabilities of sensor nodes, and we present its achievable compression gains and energy requirements in both simulation and real world experiments. Results anticipate that overall energy savings can be achieved when transferring packets of reduced sizes, even when increased CPU utilization is incurred.

1 Introduction

In general, energy budgets of nodes in wireless sensor networks (WSNs) are tightly limited [1], thus necessitating the design of applications with increased awareness to their energy consumption. As radio transmissions are an inherent and crucial characteristic of WSNs, but current radio transceivers, such as the widely employed CC2420 device, still expose power consumptions of tens of milliamperes [2], permanent operation of the radio transceiver leads to quick depletion of the battery in both transmission and reception mode. This problem can be approached in several ways, reaching from energy-aware medium access control (MAC) protocols to highly application-specific means of data compression. In this paper, we focus on compressing packet payloads, targeting to reduce

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 33–48, 2010.

the transmission duration and thus the energy required to exchange data. We investigate the achievable energy savings while disregarding the influence of MAC protocols in our analysis, as reduced packet transmission durations always correspond to savings in transmission energy. The share of the overall radio energy consumption however depends on the selected MAC protocol and its features like duty cycling and low-power listening [3]. The presented solution is designed to remain compatible with both existing header compression schemes as well as energy-aware MAC protocols. In fact, our approach is even capable of compressing both packet payloads and headers.

While data processing and compression mechanisms specifically tailored to an application may provide optimal compression results, they require individual adaptation to sensor data and packet structures and thus place an additional load on the application developer. In contrast, generic data compression solutions, as known from desktop computers, often greatly exceed the capabilities and available resources of embedded sensing systems. In this paper, we pursue the strategy to adapt a generic compression algorithm to the capabilities of sensor nodes. The resulting generic and application-agnostic solution allows to compress data without necessitating additional programming efforts. Opposed to existing approaches, which buffer multiple packets of data prior to compression, our approach targets applications that rely on immediate transmissions; i.e. each packet is compressed individually prior to its transmission.

We focus on the adaptation of a lossless adaptive data compression algorithm, based on adaptive Huffman coding (AHC), where literals in the input sequence are replaced by binary codes with a length reciprocal to the frequency of their occurrence [4]. Our analysis of the existing adaptive Huffman coder implementation for WSNs by Guitton et al. in [5] however revealed that on a TelosB platform, more than 62% of both program Flash and RAM are consumed to maintain a single compressed unicast radio connection. Instantiating more than one connection has not been possible at all due to the memory requirement for storing the corresponding Huffman code table. We address this limitation by making use of Huffman code trees with a limited number of entries, greatly reducing computational and memory consumption at the possible cost of slightly degraded compression ratios. By comparing the achievable compression gains and energy requirements, we prove the applicability and benefits of the proposed approach considering the data-oriented characteristics of traffic in many deployments.

The contributions of this paper are as follows:

1. We analyze the characteristics of WSN traffic from different deployments and prove that compression gains can be achieved when only a subset of the contained symbols are encoded.
2. We present a modification to the adaptive Huffman coding algorithm, which operates on code trees with a limited number of elements.
3. We prove its adequacy to sensor networks through an evaluation of its compression gain and energy demand as well as its applicability on real hardware.

In a first step, we present existing approaches towards data compression in WSNs in Sec. 2. We describe selected data traces taken from real sensor network deployments and estimate their compression gain when encoding only a subset of symbols in Sec. 3. In Sec. 4, we present our modifications to the AHC algorithm. Simulation results for both compression gain and energy consumption are presented in Sec. 5, followed by the results from a real-world experiment. We conclude this paper in Sec. 6 and provide an outlook on prospective future work.

2 Related Work

Pottie and Kaiser have determined in [1] that the energy demand to transfer one kilobyte of data over a distance of one hundred meters in a WSN is the same as required for executing three million CPU instructions. Later, this observation was confirmed by Sadler and Martonosi, who determined that the one-hop transmission of a single byte consumes energy equivalent to performing several thousand instructions on an MSP430 microcontroller [6]. In the same work, the authors propose the RT-LZW (retransmission LZW) algorithm, which achieves compression gains up to a factor of 2.5x when operating on aggregated data blocks of 528 bytes each. It relies on retransmissions of lost packets to ensure that data required to construct the code dictionary is present at both parties, possibly resulting in energy expenses for these additional transmissions.

Guitton et al. have analyzed the applicability of adaptive data compression in WSNs in [5]. They have extended the AHC algorithm by fault-tolerant mechanisms, which groupwise acknowledge transfers of encoded data and adapt the dictionaries to the successfully received data only. They do however not measure achievable compression gains or the energy consumption of their algorithm. When packet structures can be statically defined prior to node deployment and some fields are known to remain constant or only change incrementally, the EasiPC packet compression scheme by Ju and Cui [7] can also be used to transmit changed fields only.

In [8], Tsiftes et al. have focussed on compressing firmware updates that are transferred over the radio, and designed the SBZIP algorithm, a derivative of BZIP2, adapted to the requirements present in sensor networks. However, the implementation of SBZIP on sensor nodes does not target to compress application-generated data, but is instead used to decompress application code updates. Chou et al. present means to reduce an overall network's energy consumption by exploiting the Slepian-Wolf coding theorem in a low-complexity implementation in [9]. Hereby, no inter-node communication overhead is required as long as the correlation between the data is known. Targeting to reduce the overall number of packet transmissions, the approach is orthogonal to our concept of reducing the sizes of packets and can be used supplementary.

In [10], we have presented the Squeeze.KOM compression layer as an architectural element for sensor network nodes. Using a differential coding module, compression gains of up to 35% can be achieved at low computational cost and overall energy savings. Additionally, we have presented a feasibility study of data

compression on WSN nodes in [11]. Focused on the energy gains of application-specific compression means for a wearable sensor, we have determined overall platform energy savings of up to 5% in a realistic application setting.

We are however not aware of any previous work that discusses the energy efficiency of adaptive compression algorithms in detail while providing an extensive analysis of their applicability on current WSN hardware.

3 Analyzing the Traffic in Existing Sensor Networks

In the last decade, a variety of WSNs have been deployed in a wide range of scenarios, including wildlife surveillance [12,13], object tracking [14], or environmental monitoring [15]. In most of the WSN deployments, network traffic follows a convergecast scheme; all data is routed out of the network using a collection tree or equivalent means, rooted at one or more sinks [16]. Especially when the packet payload is comprised of environmental data, transfers often take place at a regular interval. Timely message delivery is not essential in such scenarios, but the loss of a series of packets is often interpreted as a node failure, hence regular successful transmissions are essential to determine the state of the network.

For our analysis, we have considered four exemplary data sets from existing WSN deployments: PermaSense [15], Glacsweb [17], and two series taken from the Porcupines [18]. For PermaSense, we have used 19,730 packets of 30 byte payload each transmitted by node 2036 from 15 November to 15 December 2008, taken from the project website[1]. From the Glacsweb deployment, we have used all 523 available packets of 52 byte payload, and in case of the Porcupines, we have selected two representative phases of 2.203 packets of 42 bytes each, where the first one was recorded during wearer activity (termed *activity phase*) and the second one when the wearer was asleep (*sleep phase*). While the two former data sets are physical measurements from sensors deployed for environmental monitoring, with readings changing smoothly over time, the latter are taken from motion sensors attached to a human and thus reflect both phases of sudden motions and steadiness. Representative excerpts of the four data sets are plotted in Fig. 1 for reference. It should be noted at this point that only five different symbols are present in the entire data stream in the Porcupine sleep phase, whereas the active Porcupine data set is composed of 89 different values. Glacsweb makes use of 185 different symbols, and PermaSense spans the entire input symbol range of 256 values.

To attain an estimate for the compressibility of the data sets, we show the analysis of their symbol distributions in Fig. 2, showing that the occurrence frequencies of the used symbols are not distributed evenly over the data set. In contrast, the data sets rather expose a number of subset of symbols with significantly greater occurrence numbers. The cumulative distribution function of the symbols, which is also shown in the figure, also indicates that only a fraction of the contained symbols show frequent occurrences, while the remaining symbols have almost negligible occurrence numbers.

[1] http://tik42x.ee.ethz.ch:22001

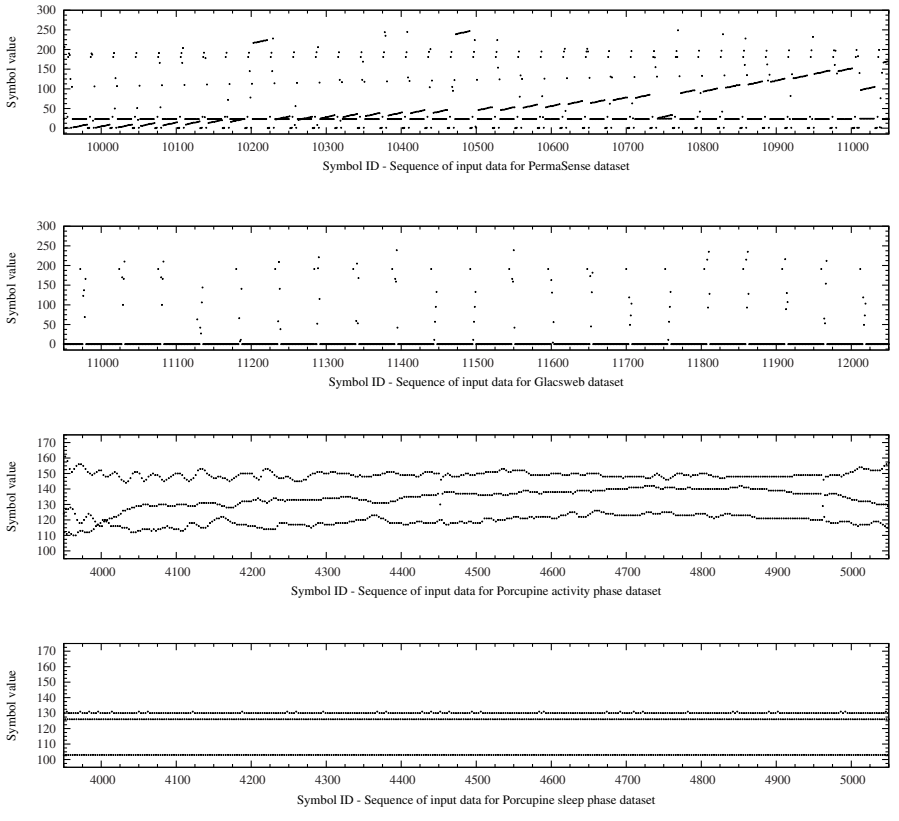


Fig. 1. Representative excerpts of the used data sets

3.1 Huffman Coding Revisited

The foundation of Huffman coding is the assignment of codes to input symbols, with their length being reciprocal to their occurrence frequency within the input stream. In static Huffman coding [19], the input sequence is analyzed prior to encoding, and occurrence frequencies of all contained symbols are determined. On completion of this process, a tree is constructed, containing mappings for all input symbols to their corresponding Huffman code. This tree must be sent to the receiver before the actual data is transmitted to ensure both parties operate on the same dictionary. This represents additional overhead, which is however generally encountered by a near-optimal adaptation to the input sequence. The major drawback when using static Huffman coding is the required full knowledge of the data, which strongly limits its applicability in sensor networks, where sensor readings become available periodically. In such case, the algorithm needs to operate on individual packets, and thus transmit the code table in each of them.

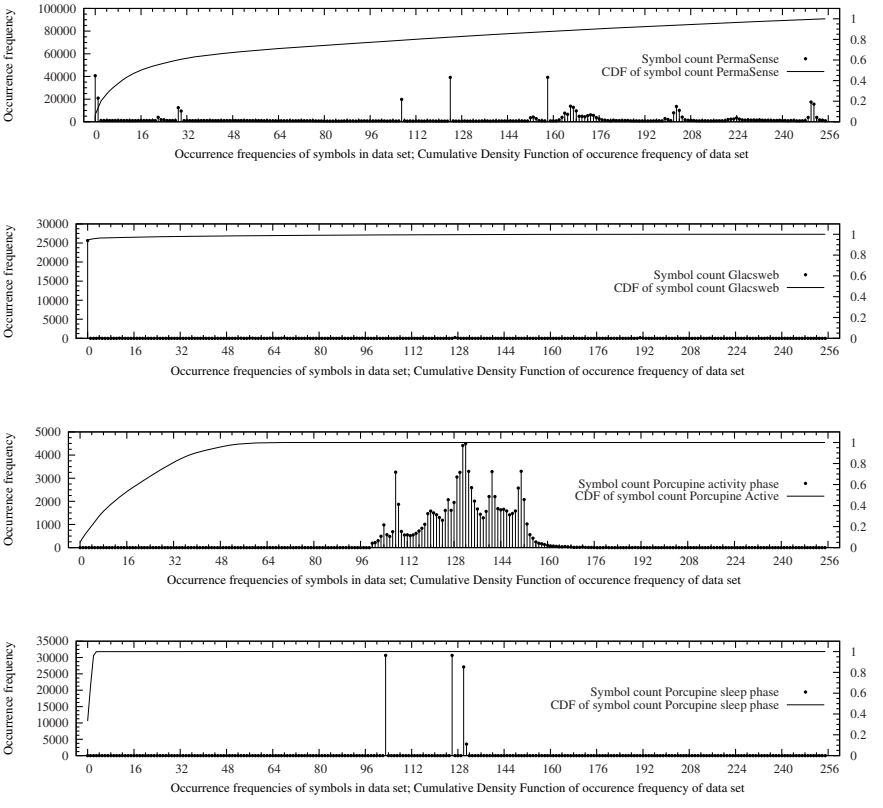


Fig. 2. Symbol distributions for the used data sets

Adaptive Huffman coding is based on the maintenance of a the code table in a dynamic way [4]. In contrast to static Huffman coding, where this table is generated prior to the actual encoding step, AHC assigns (and possibly modifies when occurrence frequencies change) the code tree during runtime. To allow for these dynamic adaptations to occur, a dedicated placeholder symbol for an input symbol not yet encountered (NYE) is part of the code tree. This symbol is always maintained with an occurrence frequency of zero and thus always assigned one of the longest codes. Whenever a symbol not yet present in the Huffman table needs to be transferred, the NYE symbol is transmitted, followed by the unencoded representation of the symbol. The symbol is then added to the code tables of both parties, so its newly assigned code can be used on its next occurrence.

3.2 Estimation of Compression Gains

In Fig. 2, the cumulative distribution functions for the studied real-world sensor data indicate that the full range of input symbols is dominated by symbols with

few occurrences within the data stream, whereas only a subset of symbols with high occurrence frequency is present. To estimate the compressibility of the data, we evaluate the resulting output sizes when only a subset of symbols is being compressed while all remaining symbols are sent unencoded.

Let us assume that a compression algorithm can encode n symbols of the size of a byte, leaving the remaining $256 - n$ symbols uncompressed. We furthermore assume that γ_i represents the number of occurrences of the byte value i in the input sequence, and that $f(i)$ is the function that assigns a code length (in bits) to this symbol. In case of an uncompressed transmission, $f(i)$ would statically be assigned a value of eight bits. Given these definitions, the length l of the output sequence resulting from the data compression step can be calculated as shown in Eq. 1, which sums the lengths of each symbol's code multiplied by the number of its occurrences within the input sequence.

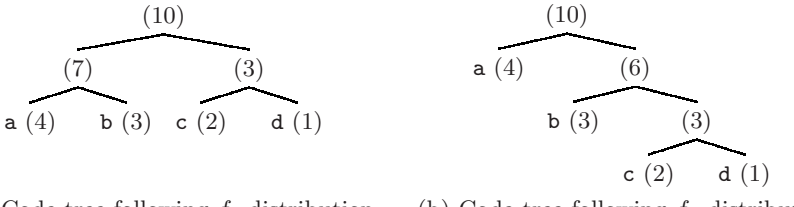
$$l = \sum_{i=1}^{256} f(i) * \gamma_i \tag{1}$$

Symbol-oriented compression schemes, such as Huffman coding, create the code length function $f(i)$ from the state of their code table. To assess if compression with a reduced number of entries in the code tree is feasible, we have used two approximation functions for code lengths; while f_e in Eq. 2 assumes an equal length for the symbols that are encoded, f_f in Eq. 3 assigns the lengths of the output codes to follow the symbol's rank $r(i)$ within the occurrence frequency list. Code trees for both functions are also depicted in Fig. 3.

$$f_e(i) = 1 + \begin{cases} \lceil ld(n) \rceil & \text{if} \quad i <= n \\ 8 & \text{if} \quad i > n \end{cases} \tag{2}$$

$$f_f(i) = 1 + \begin{cases} r(i) & \text{if} \quad i < n \\ n - 1 & \text{if} \quad i = n \\ 8 & \text{if} \quad i > n \end{cases} \tag{3}$$

When only a subset of the possible input symbols is present within the table mapping from input symbol to corresponding code, an additional indicator is required to mark the following bits as plaintext or encoded symbol. We have selected a one bit prefix to allow for this distinction, which is also reflected in the two functions. The results for this preliminary analysis are shown in Fig. 4, which



(a) Code tree following f_e distribution (b) Code tree following f_f distribution

Fig. 3. Trees for f_e and f_f with $n = 4$, resulting from the input sequence `aaaabbbccd`

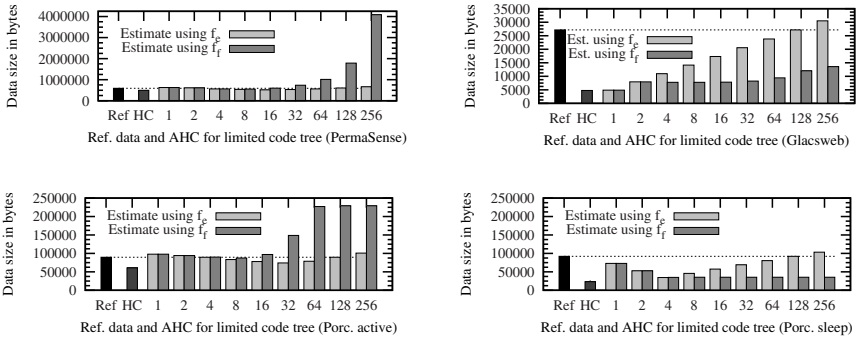


Fig. 4. Compression gain estimates for the data sets using f_e, f_f, and Huffman coding

additionally indicates the compression gains when using static Huffman coding to put the results into perspective. Although clearly indicating that savings can be achieved even when using the presented non-ideal code length distributions, the compression gain shows a strong dependence on the used data set.

As the Glacsweb and Porcupine (sleep mode) data sets only expose a small number of symbols with high occurrence frequency, the f_f function presents a better basis to achieve high compression gains, as very short codes are assigned to the most frequently occurring symbols. This way, gains of 82% are achieved for Glacsweb (at n=1), and up to 62% for the Porcupines (at n=4). In contrast, the active Porcupine and PermaSense data sets contain a larger number of frequent symbols, which are not covered well by the ranking performed in f_f. When applying f_e instead, compression gains of 17.3% (at n=32) for the active Porcupine phase, and 12% for PermaSense (at n=16) can be determined.

4 Adaptive Huffman Coding in Sensor Networks

As outlined in Sec. 3.1, a Huffman code tree must be stored for each communication link, with each of the nodes in the tree containing information about the symbol it represents, its occurrence frequency, its status (e.g., root, leaf, or NYE) as well as the identities of its children nodes and its parent. As $2n-1$ nodes are required to allow for n code entries in a tree, 511 nodes must be stored within the tree to allow for mappings of 256 input symbols. This number requires nine bits to be represented and thus two bytes on any byte-aligned microcontroller. As each tree node needs to store six bytes for its parent and child identities as well as the input symbol it represents, its frequency and status information, a minimum of nine bytes are consumed. In summary, this results in a demand of more than four kilobytes of RAM for a Huffman tree storing 256 symbols. Besides the tree itself, a table for the occurrence frequencies of input symbols must be maintained, consuming another 256 bytes at least. This theoretical analysis also confirms the behavior observed in Guitton's implementation [5], where the memory consumption of the code tree disallowed us to instantiate more than one

connection. Additionally, whenever a packet is sent or received, the Huffman tree must be updated according to its new occurrence frequency by a number of swap operations, which pose computational overhead.

The analysis of the resource demands of AHC has shown its limited applicability in WSNs due to the excessive resource demands, but also resulting from the lack of dynamic memory allocation schemes in TinyOS [20]. When operating on statically assigned memory, worst case behavior needs to be assumed for the assignment of memory during compile time, i.e. memory needs to be reserved for all symbols, including those that never occur within the input sequence.

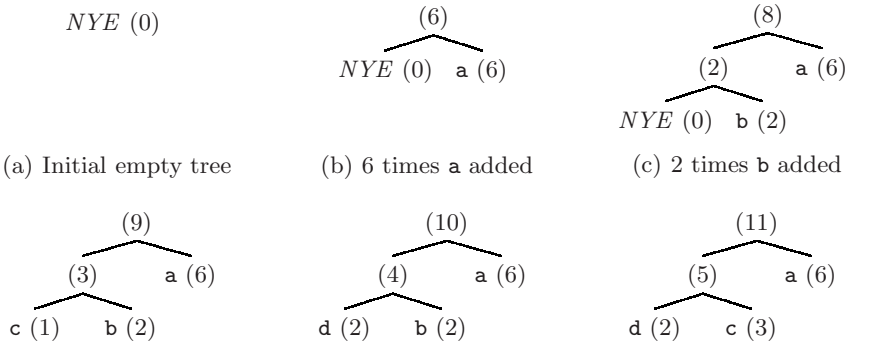
4.1 Trimming the Tree

Our observations show that the memory consumption and thus the applicability of the AHC implementation on WSN nodes is mainly limited by the number of symbols that are stored in the Huffman tree. However, as discussed in Sec. 3.2, the symbol occurrence frequencies of traffic in current WSNs are often strongly biased towards a small subset of symbols, while the remaining input characters might only rarely or never be part of the input string. Our preliminary estimations of the achievable compression gain, as shown in Fig. 4, confirm that packet size reductions are possible when only a subset of symbols are stored within the Huffman tree, while the remaining ones are transferred unencoded. The selected estimation functions were however neither adaptive to the traffic (i.e., a priori knowledge about the whole data set was required), nor did they match the characteristics of the traffic precisely.

As the memory consumption of the code table is linearly dependent on the number of entries stored within the table, keeping only a subset of input symbols in the tree can significantly reduce its memory requirement. Besides, when a smaller number of node IDs must be stored, their size can also be reduced (an 8 bit wide node ID field is sufficient to store up to 128 symbols in the tree). As a third benefit, the time to restructure the tree when changes in the occurrence frequencies are encountered also depends on the number of entries, and can in consequence be improved by reducing the tree size. In the following, we analyze the effects of confining the Huffman code tree to a limited number of entries.

4.2 Populating the Tree

The main difference between our proposed approach and conventional adaptive Huffman coding lies in the process of populating the tree. While in AHC, the NYE node is always present to attach unknown symbols to the tree, the limitation of the number of tree nodes in our algorithm can lead to situations where the NYE node, with its assumed occurrence frequency of zero, is being replaced by a symbol. We encounter this situation by keeping track of the occurrence frequencies of the symbols stored in the tree, and replacing the element with the smallest occurrence frequency in case a more frequent symbol is encountered.



(a) Initial empty tree (b) 6 times a added (c) 2 times b added

(d) 1 c added, NYE replaced (e) 2 d added, c replaced (f) 2 c added, b replaced

Fig. 5. Populating a tree with capacity for 3 symbols with the sequence aaaaaabbcddcc

We depict the operation of the proposed implementation in Fig. 5, where an input sequence of aaaaaabbcddcc and a tree capacity of 5 nodes (equalling 3 symbols) is assumed. The nodes in the tree are labeled with the symbols they represent as well as their occurrence counter. In the initial phase (Fig. 5a–c), updates to the code tree are performed identical to AHC, i.e. either the counter of a symbol present in the tree is incremented, or a new symbol is added to the tree through the NYE node. In Fig. 5(d) however, the new input symbol c is encountered in the input sequence, while the limited number of nodes disallows the NYE to create a new tree node for the symbol. In contrast to AHC, our approach replaces the NYE by the symbol node; the tree thus loses the inherent capability of being extended through the NYE node. To still adapt to the input sequence during runtime, we follow the approach of replacing the node with the smallest counter value when a symbol with greater counter is present, such as shown in Fig. 5(e) and 5(f). To allow for this, we keep track of all symbol occurrence frequencies during runtime. All resulting codes are prefixed by a single bit indicating if the following bit sequence should be interpreted as a code from the Huffman tree or as an unencoded symbol. Assuming the tree state depicted in Fig. 5(f), the letter c would thus be encoded as the binary code 101, where the 1 bit indicates that the following bits are taken from the code table, and the 01 bits refer to the branches taken to reach the value (0: left, 1: right). Similarly, symbols not contained in the table, like the numeric digit 2 can be represented as 000100010, where the first 0 bit indicates that it is followed by an unencoded symbol, and the 00100010 bits contain the ASCII representation of the digit.

The limited code tree size reduces the algorithm's resource demands significantly, as only codes for the most frequently occurring input symbols are stored, and less memory and computation time is required when reorganizing the table. Especially, as each sensor node needs to maintain a Huffman table for each connection, the proposed reduction in terms of memory consumption is essential to successfully apply AHC in WSNs. Still, the adaptive character is maintained, allowing for high compression gains.

5 Analysis and Evaluation

Concluding from the compression gain estimates presented in Sec. 3.2, it is apparent that size reductions can already be achieved when using simplified code length approximations while limiting the number of entries within the tree. In consequence, we have presented the design of an adaptive Huffman coding algorithm that operates on a limited code tree size. In this section, we analyze its compression gains when applied to the data sets introduced in Sec. 3. Secondly, we show the algorithm's applicability on sensor node hardware by evaluating both its resource and energy demands. In a third and final step, we verify the applicability of our algorithm and energy-efficiency in a real-world experiment.

5.1 Analysis of the Compression Gain

We have compressed the four presented data sets with the algorithm and varied the parameter n, indicating the number of symbols that can be stored in the tree. We show the sizes of the compressed sequences in Table 1 in comparison to the uncompressed data, which we use as reference for all following analyses.

Table 1. Output sizes in bytes (and ratio to input) for AHC with limited tree size

#Symbols in tree (n)	PermaSense	Glacsweb	Porcupines	
			active	sleep
Reference	591930 (1.0)	27144 (1.0)	89754 (1.0)	89754 (1.0)
1	625211 (1.06)	4903 (0.18)	91172 (1.02)	72816 (0.81)
2	595944 (1.01)	7929 (0.29)	88487 (0.99)	49835 (0.56)
4	567247 (0.96)	7794 (0.29)	84249 (0.94)	34504 (0.38)
8	539434 (0.91)	7766 (0.29)	79065 (0.88)	34940 (0.39)
16	517086 (0.87)	7759 (0.29)	74431 (0.83)	34940 (0.39)
32	510933 (0.86)	7772 (0.29)	70931 (0.79)	34940 (0.39)
64	519592 (0.88)	7807 (0.29)	71884 (0.80)	34940 (0.39)
128	537240 (0.91)	7869 (0.29)	71972 (0.80)	34940 (0.39)

Notably, the achievable compression gains show a strong correlation to the used data set and its characteristics. However, the number of entries in the code tree also has a major impact on the compression gain. While very small values for the symbol count n allow to encode predominant symbols in a very efficient way, the one bit prefix increases the encoded length of all other symbols. Especially in the PermaSense and active Porcupine data sets with many different contained symbols, this even leads to size increases of the output for certain configurations of n. In contrast, if too large values for n are chosen, the compression gain slightly degrades as a result of the longer code lengths of rarely occurring symbols.

5.2 Applicability on WSN Hardware

Before analyzing the algorithm's overall energy consumption, its applicability on current node hardware has been investigated. We have selected the *tmote*

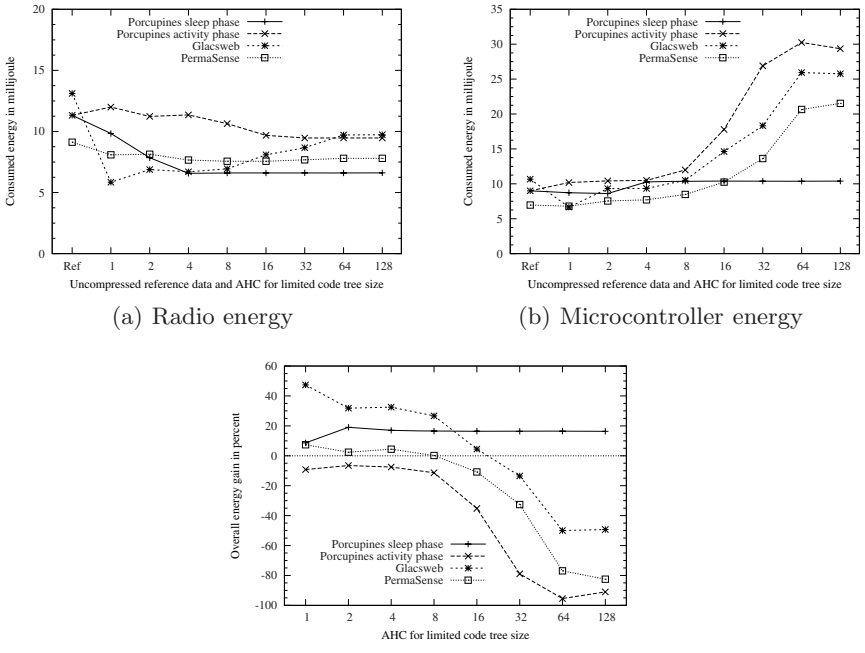
Table 2. Resource consumption of AHC with limited tree size compared to reference

#Symbols in tree	Ref	1	2	4	8	16	32	64	128	256
Flash (bytes)	22800	23838	23932	23936	23936	23936	23936	23936	23926	23918
	46.3%	48.5%	48.7%	48.7%	48.7%	48.7%	48.7%	48.7%	48.7%	48.7%
RAM (bytes)	5086	6122	6138	6170	6234	6362	6618	7130	8154	10202
	49.7%	59.8%	59.9%	60.3%	60.9%	62.1%	64.6%	69.6%	79.6%	99.6%

sky platform as our reference, comprising a TI MSP430 microcontroller (*MCU*) with 48 kilobytes of program Flash and 10 kilobytes of RAM. This platform also acts as the basis for all further analyses in this paper. To assess the resource consumption, we have implemented a simple application in the Contiki operating system [21], which periodically takes sensor readings and transmits them over the radio. We have compared our variant of the adaptive Huffman coder to the reference implementation without compression functionality. Results for the required amount of Flash and RAM are shown in Table 2 and indicate that the additional amount of resources required by our implementation stays within reasonable limits when less symbols need to be stored within the tree, even though an array containing all symbol frequencies is required. With less than an 1,150 bytes increase in the program memory consumption, and an overhead of 8 bytes per Huffman table node, the algorithm proves applicable on the used sensor node hardware, leaving sufficient resources available to the application.

5.3 Energy Analysis

If we consider the computational efforts required to process input symbols and accordingly restructure the code tree, possible size reductions of radio packets might be counterbalanced by additional expenses for the processing. To evaluate the algorithm's energy efficiency on real sensor node hardware, we have performed a detailed energy simulation using MSPsim and COOJA [22] with the corresponding NullMAC protocol implementation (i.e., the radio transceiver of the receiver node is always active, so the sender radio only needs to be switched on during packet transmissions). As discussed in Sec. 1, this particular choice of the MAC protocol has been made to evaluate the algorithm's energy demand independently of any additional effects introduced by the MAC protocol. The *sky* node type has been selected, as it also represents the platform we base our practical experiment on. To allow for reproducible results, we have statically supplied the data sets to the simulated application, and assumed a lossless wireless channel as a detailed analysis of the impact of real-world channel characteristics is beyond the scope of this paper. Assuming a single-hop transmission at a rate of ten packets per second, we have analyzed the energy requirements of the sender node only, as only marginal changes occur to the receiver's energy consumption when its radio device is not duty-cycled. We have analyzed the algorithm's energy consumption and show the corresponding results in Fig. 6. Analog to [22], we use the current consumptions measured by Dunkels et al. in [23] for our analysis. We assume an operating voltage of 3V, and radio current consumptions of



(a) Radio energy

(b) Microcontroller energy

(c) Overall energy gain

Fig. 6. Energy analysis for the adaptive Huffman coder with limited tree size

20mA in listening, 17.7mA in transmission, and 21μA in the inactive state. For the remaining platform, we have assumed 1.8mA in the active, and 5.1μA in the sleep mode.

It is evident that the use of trees with a limited number of nodes can effectively lead to reductions in the packet sizes, as observed through the reduced amount of energy spent on radio transmissions in Fig. 6(a). It can be seen that savings in radio energy of more than 50% are achieved for the Glacsweb and Porcupine sleep data sets. In case of the PermaSense and both Porcupine data sets, the reduced packet sizes lead to a consistent decrease in radio energy. Only in case of Glacsweb data, the great number of input symbols with low frequency leads to the assignment of long codes, resulting in a degraded compression ratios when larger code tree sizes are used. On the contrary, an increase in MCU utilization occurs due to the additional processing needs, as shown in Fig. 6(b). Again, the Porcupine sleep data sets exposes behavior different to the other ones, as only five symbols need to be placed in the tree. For the other data sets, a rise in the MCU energy demand is clearly visible, indicating the increased amount of energy required for for management and restructuring of the trees. The overall energy requirements, depicted in Fig. 6(c) however still prove that for the limited code tree size adaptive Huffman coder, energy gains can be observed for three of the four data sets when appropriate tree sizes, i.e. sizes in the range of 1 to 16 symbols, are chosen.

5.4 Real-World Experiment

To verify if the simulation results match the algorithm's real behavior, we have set up a real-world experiment using two tmote sky devices. The first node was configured as a sender node and supplied with the Glacsweb data set. Blocks of data were read from the Flash memory, compressed using the presented adaptive Huffman coder with limited code tree sizes, and transmitted over the radio. To limit the energy budget available to the node, we have connected its battery terminal to a boost converter powered by a supercapacitor. To allow for comparable measurements, we have put the same charge on the supercapacitor prior to each run of the experiment. A receiver node with no energy restrictions was also part of the experiment, and was used to count the number of transmitted packets in the used indoor environment. Both were configured to use NullMAC, thus allowing to compare the results to the previously performed analyses. The results of the real-world experiment with the Glacsweb data set are indicated in Table 3 and confirm that the algorithm's behavior on real hardware resembles the observed energy simulations for the given data.

Table 3. Number of packets transmitted using the AHC coder with limited tree size

#Symbols in tree	Ref	1	2	4	8	16	32	64	128
Sent packets	4733	6832	6668	6609	5991	5947	4979	4496	2581
Runtime gain	0%	44.3%	40.9%	39.6%	26.6%	25.6%	5.2%	-5.0%	-45.5%

6 Conclusion

In this paper, we have investigated the traffic characteristics of wireless sensor networks, and determined highly non-uniform symbol distributions in packet payloads; in all of our analyzed data sets, the better part of packets is comprised of a small number of different symbols only. We have shown that encoding these symbols in an efficient way, i.e. by applying adaptive Huffman coding, considerable compression gains can be achieved. To improve the applicability of existing adaptive Huffman coding algorithms on wireless sensor nodes, we have presented a lightweight version of the AHC algorithm, operating on Huffman code trees with a limited number of nodes. Our simulation results show that even when only a small number of symbols are stored in the code tree, overall energy gains can be achieved while maintaining the algorithm's applicability on sensor nodes. Our observations from a real-world experiment confirm these simulation results.

When application level data needs to be compressed, solutions that target to compress large chunks of data at a time are often unsuited for WSNs. While compression solutions for a dedicated application might allow for significant savings, they require developers to spent time and efforts on the implementation and integration. To take this burden off the programmers, we have shown that generic solutions can be designed to yield high compression ratios while being energy efficient, even when the structure of data is unknown in advance.

It is common knowledge that links in WSNs are susceptible to packet losses and variable link qualities [24]. Those issues have been addressed by existing data compression mechanisms using retransmissions [6] or fault tolerance extensions [5]. Although not directly related to the algorithm design, we plan to integrate suitable means to cope with the characteristics of real radio channels.

Acknowledgment

We would like to thank Kristof Van Laerhoven for providing more than 200 megabytes of real porcupine data, and Kirk Martinez, who supplied us with the data sets from Glacsweb. Last, but not least, our thanks go to the PermaSense project, which offers its sensor data traces for download. This research has been supported by the German Federal Ministry of Education and Research (BMBF) and the Center for Advanced Security Research Darmstadt (CASED).

References

1. Pottie, G.J., Kaiser, W.J.: Wireless Integrated Network Sensors. Communications of the ACM 43 (2000)
2. Texas Instruments Inc.: CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver, Rev. B (2007), http://www.ti.com/lit/gpn/cc2420
3. Polastre, J., Hill, J., Culler, D.: Versatile Low Power Media Access for Wireless Sensor Networks. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys (2004)
4. Vitter, J.S.: Design and Analysis of Dynamic Huffman Codes. Journal of the Association for Computing Machinery 34(4) (1987)
5. Guitton, A., Trigoni, N., Helmer, S.: Fault-Tolerant Compression Algorithms for Delay-Sensitive Sensor Networks with Unreliable Links. In: Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems, DCOSS (2008)
6. Sadler, C.M., Martonosi, M.: Data Compression Algorithms for Energy-Constrained Devices in Delay Tolerant Networks. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys (2006)
7. Ju, H., Cui, L.: EasiPC: A Packet Compression Mechanism for Embedded WSN. In: Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA (2005)
8. Tsiftes, N., Dunkels, A., Voigt, T.: Efficient Sensor Network Reprogramming through Compression of Executable Modules. In: Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON (2008)
9. Chou, J., Petrović, D., Ramchandran, K.: A Distributed and Adaptive Signal Processing Approach to Reducing Energy Consumption in Sensor Networks. In: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM (2003)
10. Reinhardt, A., Hollick, M., Steinmetz, R.: Stream-oriented Lossless Packet Compression in Wireless Sensor Networks. In: Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON (2009)

11. Reinhardt, A., Christin, D., Hollick, M., Steinmetz, R.: On the Energy Efficiency of Lossless Data Compression in Wireless Sensor Networks. In: Proceedings of the 4th IEEE International Workshop on Practical Issues in Building Sensor Network Applications, SenseApp (2009)
12. Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., Anderson, J.: Wireless Sensor Networks for Habitat Monitoring. In: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, WSNA (2002)
13. 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: Proceedings of the 10th Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS (2002)
14. Tseng, Y.C., Kuo, S.P., Lee, H.W., Huang, C.F.: Location Tracking in a Wireless Sensor Network by Mobile Agents and Its Data Fusion Strategies. In: Zhao, F., Guibas, L.J. (eds.) IPSN 2003. LNCS, vol. 2634, pp. 625–641. Springer, Heidelberg (2003)
15. Beutel, J., Gruber, S., Hasler, A., Lim, R., Meier, A., Plessl, C., Talzi, I., Thiele, L., Tschudin, C., Woehrle, M., Yuecel, M.: PermaDAQ: A Scientific Instrument for Precision Sensing and Data Recovery in Environmental Extremes. In: Proceedings of the 8th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN (2009)
16. Annamalai, V., Gupta, S.K.S., Schwiebert, L.: On Tree-Based Convergecasting in Wireless Sensor Networks. IEEE Wireless Communications and Networking 3 (2003)
17. Martinez, K., Ong, R., Hart, J.: Glacsweb: A Sensor Network for Hostile Environments. In: Proceedings of the 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, SECON (2004)
18. Van Laerhoven, K., Gellersen, H.W., Malliaris, Y.G.: Long-Term Activity Monitoring with a Wearable Sensor Node. In: Workshop on Wearable and Implantable Body Sensor Networks, BSN (2006)
19. Bentley, J.L., Sleator, D.D., Tarjan, R.E., Wei, V.K.: A Locally Adaptive Data Compression Scheme. Communications of the ACM 29(4) (1986)
20. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System Architecture Directions for Network Sensors. In: Proceedings of the 10th Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS (2000)
21. Dunkels, A., Grönvall, B., Voigt, T.: Contiki – a Lightweight and Flexible Operating System for Tiny Networked Sensors. In: Proceedings of the 1st IEEE Workshop on Embedded Networked Sensors, Emnets-I (2004)
22. Eriksson, J., Österlind, F., Finne, N., Tsiftes, N., Dunkels, A., Voigt, T., Sauter, R., Marrón, P.J.: COOJA/MSPSim: Interoperability Testing for Wireless Sensor Networks. In: Proceedings of the 2nd International Conference on Simulation Tools and Techniques For Communications, Networks And Systems, Simutools (2009)
23. Dunkels, A., Österlind, F., Tsiftes, N., He, Z.: Software-based On-line Energy Estimation for Sensor Nodes. In: Proceedings of the 4th Workshop on Embedded Networked Sensors, EmNets (2007)
24. Szewczyk, R., Polastre, J., Mainwaring, A., Culler, D.: Lessons from a Sensor Network Expedition. In: Karl, H., Wolisz, A., Willig, A. (eds.) EWSN 2004. LNCS, vol. 2920, pp. 307–322. Springer, Heidelberg (2004)

Querying Dynamic Wireless Sensor Networks with Non-revisiting Random Walks

Marco Zuniga[1], Chen Avin[2], and Manfred Hauswirth[1]

[1] Digital Enterprise Research Institute
National University of Ireland, Galway
{marco.zuniga,manfred.hauswirth}@deri.org
[2] Department of Communication Systems Engineering
Ben Gurion University of the Negev, Israel
avin@cse.bgu.ac.il

Abstract. The simplicity and low-overhead of random walks have made them a popular querying mechanism for Wireless Sensor Networks. However, most of the related work is of theoretical nature and present two important limitations. First, they are mainly based on simple random walks, where at each step, the next hop is selected uniformly at random among neighbors. This mechanism permits analytical tractability but wastes energy by unnecessarily visiting neighbors that have been visited before. Second, the studies usually assume static graphs which do not consider the impact of link dynamics on the temporal variation of neighborhoods.

In this work we evaluate the querying performance of Non-Revisiting Random Walks (NRWs). At each step, NRWs avoid re-visiting neighbors by selecting the next hop randomly among the neighbors with the minimum number of visits. We evaluated Pull-only and Pull-Push queries with NRWs in two ways: (i) on a test-bed with 102 tmotes and (ii) on a simulation environment considering link unreliability and asymmetry. Our main results show that non-revisiting random walks significantly improve upon simple random walks in terms of querying cost and load balancing, and that the push-pull mechanism is more efficient than the push-only for query resolution.

1 Introduction

Querying has been, and continues to be, one of the most investigated areas in the Wireless Sensor Networks community. For scenarios where nodes have no location information (location-less), querying paradigms can be classified into 2 broad categories: i) random walks [20,5,21] and ii) flooding or controlled flooding (expanding ring searches) [11,12,13].

On flooding, each node (re)transmits the querying packet once. On random walks, nodes are queried in some sequential random order. The walk starts at some fixed node, and at each step it moves to a neighbor of the current node. The random walk is called simple when the next node is chosen uniformly at random from the set of neighbors.

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 49–64, 2010.

The main advantage of random walks is its localized search, which avoids the unnecessary use of bandwidth and energy resources utilized by flooding-type techniques [17]. On the other hand, if the data of interest is far away from the sink, the querying cost of random walks can be super-linear in the worst case compared to the linear cost of flooding.

In this work, we investigate a variant of random walks that provides an energy-efficient querying alternative for location-less deployments: Non-Revisiting Random Walks (NRWs)[1]. The motivation behind this work is to derive a querying mechanism that combines the localized behavior of random walks and the linear cost of flooding.

Our work is inspired by the studies presented in [6,22]. These studies identify a important limitation of Simple Random Walks (SRWs): selecting the next node at random is a simple mechanism but leads to frequent revisiting nodes, which in turn leads to long delays and high expenditures of energy. Contrary to the *blind* selection performed by simple random walks, NRW selects the neighbor with the least number of visits. This Non-Revisiting mechanism maximizes the likelihood of encountering unvisited nodes, and hence, accelerates the discovering process.

Our work focuses on two types of querying scenarios: (i) Pull-Only querying and (ii) Push-Pull querying. In Pull-Only querying the sink starts a walk looking for the event. In Push-Pull querying, both, the event and sink nodes start walks and query is solved when the walks intersect.

We evaluated the performance of SRW and NRW on TWIST [3], an in-building test-bed with 102 tmotes, and we simulated larger networks using a probabilistic link model for the channel [28]. Our results provide two important contributions. First, it illustrates the difficulties faced by random walks on real deployments due to the high temporal dynamics of links. We show that polling the neighborhood immediately before transferring the token is an efficient mechanism to cope with these dynamics. Second, our results indicate that NRW, together with the simple push-pull mechanism, is an efficient querying mechanism for networks consisting of up-to thousands of nodes. NRWs with Push-Pull querying maintains the elegance of simple random walks, while at the same time provide querying costs that are linear or sub-linear (depending on the size of the network).

2 Definitions, Implementation and Metrics

First, let us present the precise definitions of the random walks types and the querying mechanisms evaluated on our work:

Definition 1 (Simple Random Walk (SRW)). *The walk starts at an initial node and at each step selects one of its neighbors uniformly at random.*

Definition 2 (Non-Revisiting Random Walk (NRW)). *The walk starts at an initial node and at each step selects the neighbor with the minimum number of*

[1] There are similar types of walks in the related literature (i.e., self-avoiding walks [15], Vertex- Reinforced Random Walks [18]), but to the best of our knowledge NRWs where not considered explicitly before.

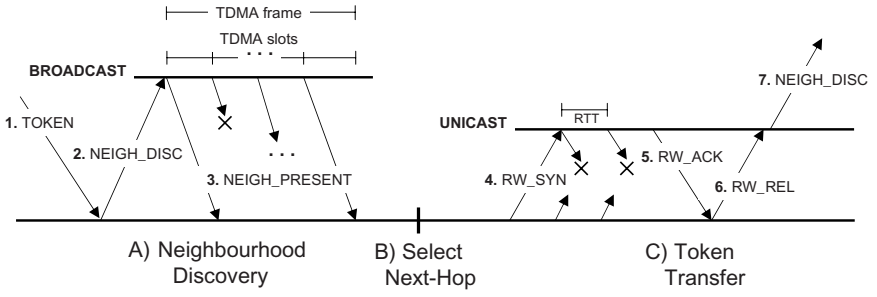


Fig. 1. Protocol Implementation of Random Walk

visits (which could be 0). If more than one node have the same minimum number of visits, the next node is selected uniformly at random among these nodes.

We consider two mechanisms for query resolution. In both of them the *event-node* has some data of interest, and the *sink-node* issue a query to find that piece of data.

Definition 3 (Pull-Only Querying). *The data remains on the event-node and only the sink-node starts a random walk (i.e., pull). The query is solved when the walk reaches the event node.*

In the push-pull case the event-node *publishes* its data.

Definition 4 (Push-Pull Querying). *The event-node starts a random walk to publish its data of interest (i.e., push). The sink-node starts a random walk based query (i.e., pull). The query is solved when the paths of the walks intersect.*

We do not discuss here the way the data of interest is routed back to the sink after query resolution, but this could be done for example by using a trace left by the query walk. For the reminder of the paper, the term *token* is used to denote the *presence* of the walk on a node.

2.1 Walk Implementation

Contrary to theoretical studies, where the neighborhood of a node is assumed to remain constant, in real scenarios, link dynamics such as asymmetry, unreliability and temporal variation pose significant challenges to the robust dissemination of the walk. In order to cope with these dynamics, our implementation of a random walk utilizes the following three procedures: (a) Neighborhood Discovery, (b) Selection of Next-Hop and (c) Transferring of Token. These procedures are presented in Figure 1.

Neighborhood Discovery. Upon reception of the token, a node broadcasts a `NEIGH_DISCOVERY` message. Nodes within the transmission range of the sender reply with `NEIGH_PRESENT` messages. In order to avoid collisions caused by the

concurrent transmission of `NEIGH_PRESENT` messages, we implemented a MAC TDMA scheme. In this TDMA scheme, nodes are assigned different transmission slots based on their *id*.

Selection of Next Hop. The token-holder waits until the end of the TDMA frame and selects the next node among the received `NEIGH_PRESENT` messages. Depending on the type of walk to be performed, the selection follows the guidelines presented in Definitions 1 and 2.

Transferring of Token. This procedure is similar to the 3-way handshake mechanism utilized in the TCP protocol. The token-holder sends an initial `RW_SYN` packet to communicate a node that it has been selected as the next step. Upon reception of a `RW_SYN`, the receiver sends a `RW_ACK` packet. Finally, the sender completes the transfer by sending a `RW_REL` packet. In order to cope with packet losses, `RW_SYNs` and `RW_ACKs` are sent every `RTT` (round trip time). The sender stops transmitting `RW_SYNs` after receiving a `RW_ACK`, and the receiver stops transmitting `RW_ACKs` after receiving a `RW_REL`. `RW_REL` packets are sent only upon reception of a `RW_ACK`.

2.2 Metrics

In this subsection we present the metrics used to quantify the performance of SRW and NRW. Let us denote G_n as the communication graph formed by a network of n nodes and s as the number of steps performed by a random walk. Based on this notation, a simple random walk performing s steps on graph G_n is denoted by $SRW(G_n, s)$, and a non-revisiting random walk is denoted by $NRW(G_n, s)$.

Once a random walk starts, each node $u \in G_n$ stores locally the following information:

- T_u^{min}: time of first visit.
- T_u^{max}: time of last visit.
- V_u: number of visits.

In our work, the time t is represented by the number of steps. For example, a node u that is visited for the first time at the k^{th} step of the walk will have an entry $T_u^{min} = k$.

Two important properties of random walks are directly related to querying [16]: (i) cover time and (ii) hitting time. The cover time $C_u(G_n)$ is the expected number of steps for a walk starting at u to visit all the nodes in graph G_n. The *partial cover time* $C_u(G_n, f)$ is the expected number of steps for a walk starting at u to first visit a fraction f of the graph G_n. The hitting time h_{uw} is the expected time taken by a walk starting at u to reach w for the first time. In this paper we evaluate the *average* hitting time $H_u(G_n)$ from a sink u which is given by:

$$H_u(G_n) = \frac{\sum_{w \in G_n} h_{uw}}{n-1} \tag{1}$$

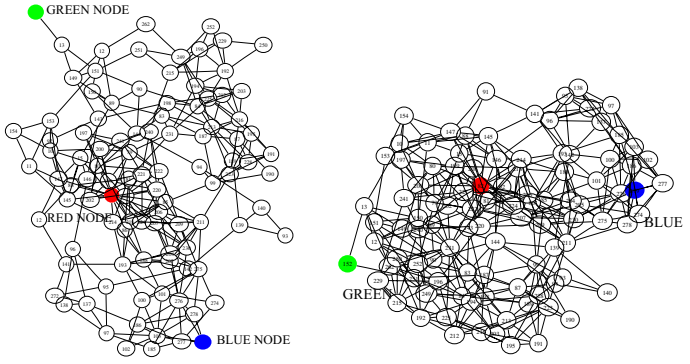


Fig. 2. Communication Graph of TWIST. (a) shows links with *transmission probabilities* greater than 0.9 and (b) greater than 0.7. Three nodes were selected to inject the random walks (green, red, blue).

Hence, for Pull-Only querying, cover time and hitting time translate to the worst-case and average-case querying scenarios[2]. Another important property of random walks is load balancing. Given the limited energy resources of WSN, it is desirable that the walk visits the network evenly without over-stressing some nodes by visiting them more frequently. For a starting node u, we measure the load balancing as the difference between $B_u^{\max}(G_n, s)$ and $B_u^{\min}(G_n, s)$, the expected maximum and minimum (respectively) number of visits observed by nodes in the network after s steps. Denoting $V_i(s)$ as the number of visits on node i after s steps, formally:

$$B_u^{\max}(G_n, s) = E\left[\max_{i \in G_n}\{V_i(s)\}\right] \tag{2}$$

$$B_u^{\min}(G_n, s) = E\left[\min_{i \in G_n}\{V_i(s)\}\right] s.t.\ V_i(s) > 0 \tag{3}$$

3 Experimental Results: Medium-Scale Networks

3.1 Testbed and Experiment Setup

The simple and non-revisiting random walks were implemented in TinyOS 2.0.2 and evaluated on TWIST [3]. TWIST is a remote wireless sensor network test-bed deployed on a building and it has 102 tmotes. The nodes are not mobile, hence, the dynamics observed on the links are due to the surrounding environment.

[2] Additional important measure is the *maximum hitting time* which is the maximum over all h_{uw}, it will be considered in future work.

We utilized the lowest output power available on tmotes (-25 dBm)[3]. Figure 2 shows the communication graph of the network for (a) links with transmission probability above 0.9 and (b) above 0.7. The location of the nodes in the graph is not represented by their actual physical coordinates, but rather, by virtual coordinates obtained with Graphviz [1] based on the connectivity matrix.

We selected three nodes as the starting points for the walks (green, red and blue nodes). These nodes were selected to capture approximately the diameter and radius of the graph. For the remainder of the paper we denote these nodes by g, r and b, respectively. On each one of these three nodes we injected 10 simple random walks and 10 non-revisiting random walks, that is, a total of 60 walks were performed. Each walk was assigned a different random seed and it performed 1000 steps.

First, we present results concerning the temporal variance of the neighborhoods caused by link dynamics. Then, we present results for Pull-Only and Push-Pull querying.

3.2 Link Dynamics

Theoretical studies of random walks do not capture the impact of temporal dynamics on the total transmission costs incurred by the network. Most of these studies are done under ideal conditions that assume a constant neighborhood for all nodes throughout the network lifetime. Unfortunately, node failures, channel multi-path, dynamic environments and other factors lead to highly dynamic neighborhoods in WSN. In order to filter out links affected by these temporal dynamics, our implementation polls a node's neighborhood immediately before transferring the token (Neighborhood Discovery phase in Section 2.1)[4].

In this subsection, we show that *link asymmetry*and *neighborhood variance* are important challenges faced by random walks in WSN. We also show that the Neighborhood Discovery phase is a simple yet robust and efficient mechanism to cope with these dynamics.

Asymmetric Links. Link asymmetry refers to the phenomena where a node A can communicate with node B, but node B can not communicate with node A. Several works [28,10,9] have shown that asymmetric links are pervasive in WSN. Link asymmetry presents a serious inconvenience for random walks because bidirectional links are required to transfer the token at each step. In order to capture link asymmetry, at each neighborhood poll, we evaluated the difference between the number of nodes receiving the `NEIGH_DISC` packet and the number of `NEIGH_PRESENT` messages received at the sender. A neighborhood poll has 0-degree asymmetry if it reports bidirectional links with all neighbors, i.e. the token-holder receives `NEIGH_PRESENT` packets from all neighbors that received

[3] Utilizing higher output powers leads to graphs with high densities and short diameters. Graphs with these characteristics are not challenging querying scenarios.

[4] A different approach would be to poll neighbors when the network start functioning [4] or on a periodic basis. However, the neighborhood information of these mechanisms could quickly become inaccurate due to the high temporal variance of links.

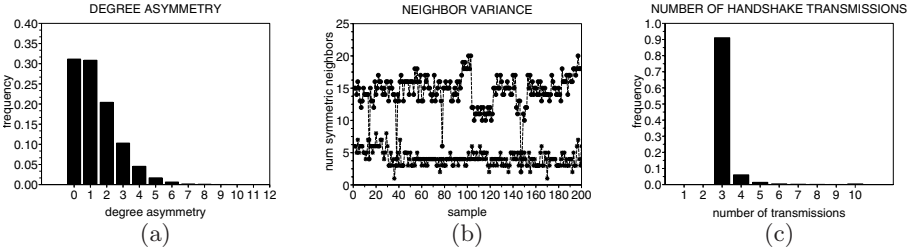


Fig. 3. Impact of link dynamics on (a) Degree Asymmetry and (b) Neighborhood Variance. Utilizing the Neighborhood Discovery phase limits the packet losses during the transfer of the token, as shown on (c).

the `NEIGH_DISC` packet. A neighborhood poll has x-degree asymmetry ($x > 0$) if it observes x asymmetric links in its neighborhood, i.e. there are x neighbors that received the `NEIGH_DISC` packet but their `NEIGH_PRESENT` replies were lost.

Figure 3 (a) depicts the results for 30000 neighborhood polls (approximately 300 polls performed by each node). We observe that only 30% of neighborhood polls observe purely symmetric links. Had the Neighborhood Discovery phase been performed only once (at the beginning of the process), some of the asymmetric neighbors would have been used in futile attempts to transfer the token.

Neighborhood Variance. Filtering asymmetric links is a necessary but insufficient step to cope with link dynamics. Symmetric links also have high temporal dynamics. Effects such as node failures and movements in the surrounding environment lead to intermittent links. These intermittent links affect significantly the neighborhoods observed by the nodes. We denote this intermittent phenomena as Neighborhood Variance. Figure 3 (b) captures the dynamics of the topology. This figure shows the number of *bidirectional* neighbors observed by two nodes at different instants of time (samples), one node with a high average degree and the other with low average degree. Clearly, the temporal dynamics observed by the nodes is significant – similar dynamics are observed for all nodes in the network. By providing an accurate representation of the available bidirectional neighbors, random walks can conduct a more-informed selection of the next step.

Number of Handshake Transmissions. The unreliable nature of WSN links requires a 3-way handshake mechanism to transfer the token reliably at each step. In order to minimize communication costs, it is desirable to use as few transmissions as possible at each step. Figure 3 (c) demonstrates the value of the Neighborhood Discovery phase. By filtering asymmetric links and intermittent bidirectional links, we avoid a potentially large number of packet losses during the transfer of the token. 90% of transfers utilize the minimum number of transmissions required (3). Furthermore, most transfers (>99%) are achieved with 6 transmissions or less (at most three packet losses during the handshake process). These reliable 3-way transmissions are obtained due to the temporal correlation in link quality [10,24]. A good link at time t is likely to still have a good quality at time $t + \delta$, but no accurate link quality estimation can be made

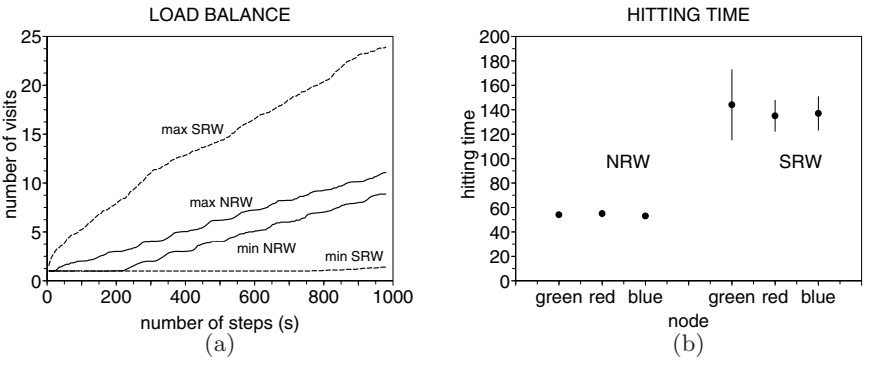


Fig. 4. (a) Load Balance and (b) Hitting Time of NRW and SRW. NRW outperforms SRW on both metrics.

for $t + \Delta$ (where $\Delta \gg \delta$). Hence, identifying reliable bidirectional links during the Neighborhood Discovery phase guarantees to a large extent the stability of the links during the token transfer.

3.3 Pull-Only Querying

In this subsection, we present results for our properties of interest in a Pull-Only querying scenario.

Load Balance. Due to the limited energy resources of WSN, it is important to distribute the energy consumption evenly across the network. Denoting $B_r^{\max}(s)$, $B_g^{\max}(s)$ and $B_b^{\max}(s)$ as the average of number of visits to the most visited node during the 10 SRWs of length s started at the red, green and blue nodes, we computed the average visits to the most visited node $B_{\text{srw}}^{\max}(s) = (B_r^{\max}(s) + B_g^{\max}(s) + B_b^{\max}(s))/3$ at each step $s = 1, \ldots, 1000$. The average number of visits to the least visited node in the SRW $B_{\text{srw}}^{\min}(s)$, and $B_{\text{nrw}}^{\max}(s), B_{\text{nrw}}^{\min}(s)$ for NRWs were computed in a similar way. As a measure of load balancing we consider the difference between the most and least visited nodes. Figure 4 (a) presents the visits to the most and least visited nodes in SRW and NRW. For example, when the number of steps $s = 400$, the least visited node on SRWs has on average 1 visit while the most visited node has on average 13 visits. For NRWs, the min and max averages are 3 and 5 respectively. This implies that NRWs do a significantly better job in distributing the use of energy resources. Furthermore, as the number of steps increase, SRWs continues to degrade, while NRWs keep the maximum and minimum number of visits within linear bounds (even distribution of load).

Hitting Time. In Pull-Only querying, the hitting time represents the expected time required to find an event that appears uniformly at random in any node of the network. For each node r, b, g, we computed the average hitting time and standard deviation for the 10 SRWs and 10 NRWs started at these nodes.

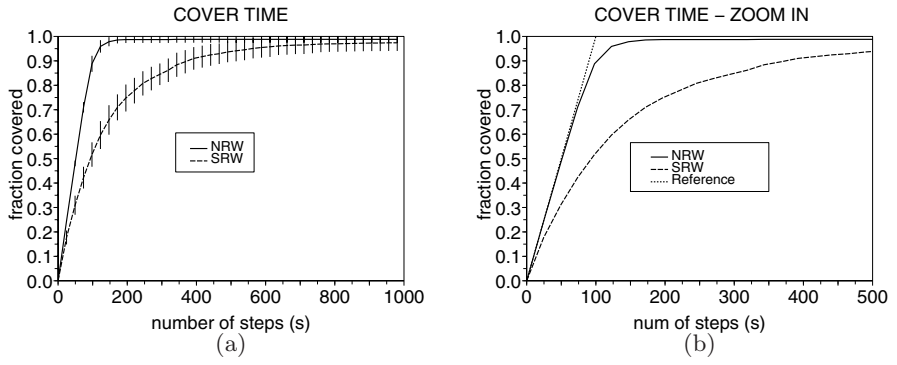


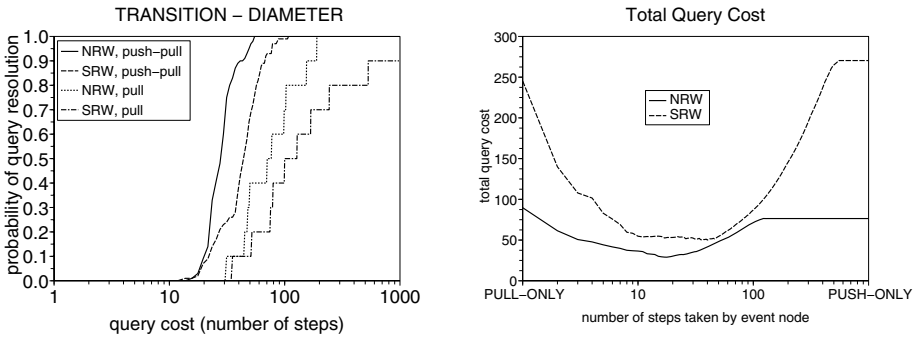
Fig. 5. Cover time of NRW and SRW. NRWs have faster cover times and less variance. NRWs also have linear partial cover times (up to approximately 70%).

Figure 4 (b) shows the results. The first three points represent NRWs and the next three points represent SRWs. There are two important observations to highlight. First, NRWs take approximately three times less steps than SRWs to solve the average query. Second, the variance among the 10 NRWs is almost negligible, but it is significant in SRW. Hence, NRWs are not only a faster and more energy efficient querying mechanism, but also provide less uncertainty.

Cover time. Several WSN scenarios require the estimation of the worst-case querying cost. When the data is of vital importance and it is not duplicated, or the query computes a function of all nodes, it may be necessary to visit (cover) all nodes in the network. Figure 5 presents the test-bed results for cover time. The SRW and NRW curves represent the average and standard deviation of 30 walks each (10 walks for each r, b and g node). In Figure 5 (a) we observe that NRW has two important advantages over SRW. First, NRW covers the network significantly faster than SRW. For instance, when $s = 100$, NRW covers 90% of the network while SRW covers 50% of the network. Second, the standard deviation of NRW is significantly lower than SRW, which leads to less uncertainty in the result of the querying process. Figure 5 (b) is a zoom-in of Figure 5 (a) and it shows that the partial cover time is linear for up to about 80% of the network (the dashed line has slope 1). The linear partial cover time indicate that most queries can be solved in linear time for NRWs.

3.4 Push-Pull Querying

The results presented in the previous subsection assumed that the events are not published (pull-only). However, several works in WSN have shown that push-pull querying mechanisms [7,14] can perform significantly better than pull-only querying. In push-pull querying, both, the sink and event inject walks and the query is solved when the two walks cross. In this subsection, we evaluate the performance of NRW in push-pull querying scenarios. The basic idea of gaining from a push-pull scheme is based on the following property.



(a) (b)

Fig. 6. (a) Probability of query resolution vs. query cost. Push-Pull has a significantly better performance than Pull-Only. (b) Total query cost. Both SRW and NRW have an optimal Push-Pull performance in-between the Push and Pull extremes. In general NRW with Push-Pull provides the best performance.

Property 1. *A sufficient condition for two random walks to intersect on a graph is that each walk visits at least* $\lceil \frac{n}{2} \rceil + 1$ *different nodes, where* n *is the number of nodes in the graph.*

Moreover, based on what it is known as the *birthday paradox*, it can be shown that two walks can intersect with high probability even in a sub-linear time:

Property 2 ([14]). *Two random walks on a graph will cross with high probability when each walk visits a uniform sample of* $O(\sqrt{n})$ *nodes, where* n *is the number of nodes in the graph*[5].

Considering the above properties and the observation that the partial cover time of NRWs is linear up to a fraction well-beyond 50% of the network (Figure 5); then, by starting NRWs at the sink and the event nodes with a maximum number of steps s_{max} around $0.5n$, there is a high likelihood that the walks will cross and solve the query.

Query Resolution Transition. In order to evaluate the performance of push-pull querying, we obtain crossing-times from the walks collected in our experiments. Considering that each node injected 10 NRWs, we evaluated the 10×10 possible combinations of walk-pairs.

The first scenario we considered is the following. Each walk was set to perform a maximum of s_{max} steps, where s_{max} takes discrete values between 1 and 1000. The event-node starts a push (publish) walk and runs until it takes s_{max} steps or stops earlier if the sink-node is found. If the sink is not found, the event-trace

[5] The time to visit a uniform sample of $O(\sqrt{n})$ nodes depends on the *mixing time* [16] of the random walk which we don't study here.

remains alive. At a later time, the sink-node starts a pull (query) walk[6] and stops when it hits the trace left by the event-walk, otherwise, the sink-walk runs until completing s_{max} steps. Let s_{sink} and s_{event} be the number of steps taken by the sink and event walks, and $s_{total} = s_{sink} + s_{event}$ be total number of steps required to solve a query (the query cost).

Figure 6 (a) presents the cumulative distribution function *cdf* of the query resolution cost for SRW and NRW. For completeness, we also provide the *cdf* for pull-only querying[7]. The curves for push-pull are actually *lower bounds* for the probabilities of query resolution (since we use $2s_{max}$ as the query cost). In practice, the total query cost is much smaller than $2s_{max}$ (as we will show later).

In general, Push-Pull querying provides an order of magnitude better performance than Pull-Only querying for SRW and NRW. Figure 6 (a) shows the *cdf* for the diameter of the network (green and blue nodes) - approximately 8 hops[8]. For example, we observe that for a query cost of 60, the SRW pull-only solves the query with probability 0.2 and the NRW pull-only with probability 0.4. On the other hand, for the push-pull the SRW solves the query with probability about 0.85 and the NRW with probability 1.0. In Section 4, we will observe that as the size (diameter) of the network increases, NRWs increase their comparative performance with respect to SRWs.

In order to complete the test-bed evaluation of push-pull querying, we consider a second scenario. In this case the event-node issues a push walk of increasing lengths s_{event}. For a given event-node walk of length s, if it didn't reach the sink-node, *the sink-node issue a pull walk that continues to step until it crosses the event walk*. The length of the sink walk is denoted s_{sink}. The total query cost is $s_{total} = s_{event} + s_{sink}$ and we then evaluate the average total query cost for each s. Note that when $s = 0$, the query is pull-only and when s is very large the query is push-only[9]; for other values of s the query is push-pull.

Figure 6 (b) shows the average query cost for SRW and NRW for increasing push walk lengths. Our two main observations are validated again here. First the NRW solves the query in less steps than the SRW. Second the push-pull query resolution is more efficient in terms of number of steps than the pull-only or push-only queries. The data shows that NRW optimal query cost is about 29 steps when s_{event} is 17 steps, while the SRW cost is about 52 for 17 steps and about 50 at the optimum when s_{event} is 40. More generally, the optimum query cost seems to be when s_{event} and s_{sink} are about the same size.

4 Simulation Results: Large-Scale Networks

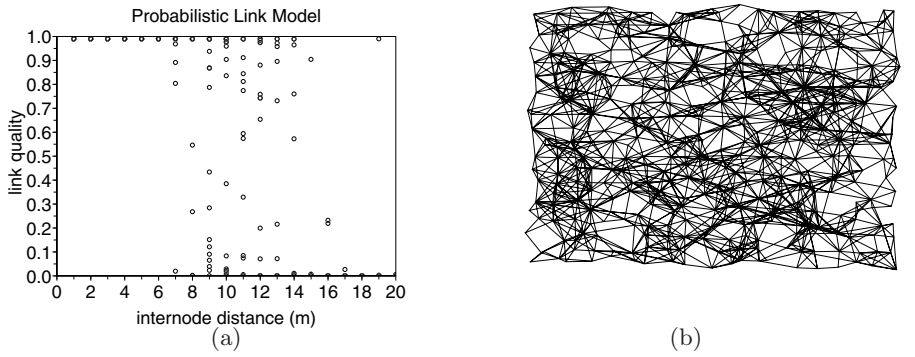
The test-bed results provide interesting empirical observations, but these results are confined to the particular size (102 motes) and characteristics of the TWIST

[6] The focus of this section is on crossing-times, and hence, we assume that the query walk is started within the lifetime of the event-trace.

[7] For pull-only queries, we utilize the 10 empirical walks available at each node. Due to these limited number of walks, the curves show the staircase form.

[8] A deterministic calculation of the diameter is not possible due to link dynamics.

[9] These costs are not necessarily equal since hitting times are symmetric.



(a) (b)

Fig. 7. Link Probability Model. (a) samples of link quality vs. distance. (b) sample of a network with 400 nodes and output power -10 dBm.

network . In order to validate the results for larger networks we perform simulations on WSN topologies that include link unreliability and link asymmetry. It is important to remark that these simulations capture some degree-heterogeneity due to multi-path channels and hardware variance, but they do not capture temporal variance. Hence, the main motivation of the simulations is to observe if the partial cover time of NRWs remain linear for larger networks.

4.1 Simulation Environment

We performed simulations using Scilab [2], an open-source alternative to Matlab.

Topology. Various network sizes were tested (100, 400, 900 and 1600). The network followed a normal-random topology, where nodes are initially deployed on a regular grid layout with an internode distance of $d = 5$ meters. Then, a 2-D normal r.v. is used to introduce a perturbation on the x and y coordinates of each node. The idea of a more uniformly distributed topology, compared to a pure random deployment, was borrowed from Glomosim [26].

Communication Model. The link quality among nodes was calculated based on the probabilistic model presented in [28]. This model captures unreliable and asymmetric links and it is given by:

$$p(d) = (1 - \frac{1}{2}\exp^{-\frac{\gamma(d)}{2}\frac{1}{0.64}})^{8f} \tag{4}$$

$p(d)$ is the link quality for an internode distance d. f is the number of bits transmitted and $\gamma(d)$ is the signal to noise ratio, which includes the output power and channel parameters. In our simulations f=160 bits and the channel parameters are 3.0 for both, the path loss exponent and the shadowing variance [23,28]. Figure 7 (a) shows samples of link quality for various internode distances; as we observe, the model resembles the behavior of empirical studies [27,25,28].

Simulations Run. We utilized an output power of -10 dBm. The output power and channel parameters presented above aim to recreate, to some extent, graph

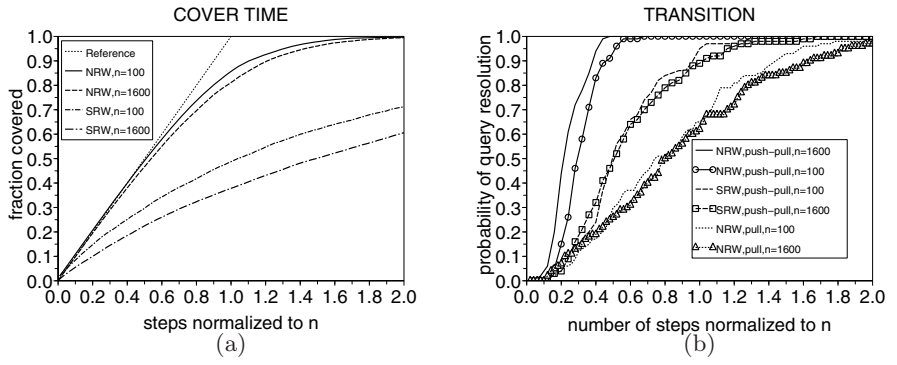


Fig. 8. Simulation Results (a) Cover time: partial cover time presents linear behavior. (b) Probability of query resolution: NRW with Push-Pull provide the best performance.

characteristics of TWIST such as degree distribution. Figure 7 (b) shows a sample topology with 400 nodes (only links with link quality > 0.7 are shown). The sink is assumed to be the node at the bottom-left corner of the graph. For Pull-Only querying, we performed 100 SRWs and 100 NRWs on each network size n. For the Push-Pull scenario, the event node was located at the top-right corner of the graph and we also run 100 SRWs and 100 NRWs starting at this node.

4.2 Simulation Results

Pull-Only. Due to space constraints, we focus on the results for cover time. The hitting time shows the same trend as the empirical results: NRW performs significantly better than SRW and the difference in performance increases in favor of NRW as the network size increases. Figure 8 (a) shows the cover time normalized to the size of the network n. In the interest of clarity, we plot results only for n=100 and n=1600 (n=400 and n=900 are in-between these curves).

The most important observation is that the partial cover times of the empirical and simulation results have the same trend: an initial long linear behavior Furthermore, once normalized, there is not a significant difference among partial cover times of NRWs for networks with different sizes. These results indicate that for larger networks NRW are also expected to solve most Pull-Only queries in linear time. For SRW, the partial cover times remain significantly longer than NRW, and the cover time degrades as the size of the network increases.

Push-Pull. In order to evaluate the effectiveness of the push-pull mechanism, the sink and event nodes were located at opposite extremes of the topology (diameter of graph). First, the node at the top right corner (event) started the walk for s steps, and then, the node at the bottom left corner (sink) started the walk. For completeness, the figure also shows the pull-only performance of NRWs. Figure 8 (b) depicts the *cdf* of query resolution for $n = 100$ and $n = 1600$. We observe that the Push-Pull with NRWs provides the best performance, followed by Push-Pull with SRWs and finally Pull-Only with NRWs. The results for

Pull-Only with SRWs is not shown but the probability of solving the query after $s = 2n$ was less than 0.4 for all networks' size. Also, the difference in performance between Push-Pull NRWs and Push-Pull SRWs increase for larger networks, however this improvement is not clearly observed in the figure.

5 Related Work

The research work on querying can be classified in two main groups: location-less and location-based. In location-less deployments, nodes have information only about their neighbors presence. The most notable querying paradigms are: flooding, expanding ring searches (controlled floods) and random walks. In location-based deployments, nodes also have location information about their neighbors. This information is very useful for geographic routing and geographic hash tables. In this work we focus on random walks on location-less scenarios.

Random walks on graphs have been studied mathematically, and there is a growing body of theoretical literature on the subject [8,16]. In the context of location-less wireless sensor networks, different variants of random-walk-based protocols have been proposed and analyzed. In one of the earlier works, Servetto and Barrenechea [20] proposed and analyzed the use of constrained random walks on a grid to improve the load-balanced routing between two known nodes. In [5], the authors argue that even simple random walks can be used for efficient and robust querying because their partial cover times show good scaling behavior. The ACQUIRE protocol [19] combines random walks with controlled floods and show that this hybrid mechanism can outperform flooding and even expanding-ring-based approaches in the presence of replicated data.

The evaluation of push-pull mechanisms was inspired by important related work. Rumor routing [7] advocates the use of multiple random walks from the events as well as the sinks, so that their intersection points can be used to provide a rendezvous point. On the same line of work, Shakkottai [21] analyzed different variants of random-walk-based query mechanisms and concludes that source and sink-driven sticky-searches (similar to rumor routing) provide a rapid increase of query success probability with the number of steps. Friedman *et. al.* [14] offered and evaluated via simulation probabilistic quorum systems that use different push-pull mechanism including simple and self-avoiding random walks. Contrary to the studies mentioned above, we consider the number of visits as an important parameter to *guide* the random walk.

Our work on NRWs is mainly motivated by [6,22]. These studies show in different ways that simple random walks lead to energy wastage due to their *blind* (re)visiting mechanism. In [6], instead of selecting only one node at random, the authors propose to select two (or more) nodes at random and select the one with the minimum number of visits as the next hop. In [22], the authors use homophyly and degree information to navigate the network, and the walk "ignores visited neighbors if there is at least one unvisited neighbor".

Based upon notable contributions on random-walk-based querying, we propose and analyze Non-Revisiting Random Walks with Push-Pull querying; a simple and efficient querying paradigm for practical WSN deployments.

6 Conclusions

In this work we evaluated the performance of Non-Revisiting Random Walks (NRW). Contrary to the blind selection performed by simple random walks, NRWs select the neighbor with the minimum number of visits. This mechanism increases the likelihood of encountering unvisited nodes, and as a consequence, provides a faster coverage.

We evaluated NRWs on (i) a test-bed consisting of 102 motes and (ii) with simulations on topologies consisting of unreliable and asymmetric links. Our results provide two important contributions. First, polling the neighborhood at each step of the walk is an efficient mechanism to cope with temporal link dynamics. This polling mechanism permits an accurate representation of the neighborhood, which allows a robust token-transfer and a well-informed selection of the next steps. Second, NRWs together with a simple push-pull mechanism are an efficient querying mechanism. NRWs maintain the elegance and simplicity of simple random walks, while at the same time can provide querying costs that are liner or sub-linear (depending on the size of the network).

In this work we considered only the cost of finding the data of interest (query), but not the cost required to transfer the information back to the sink. In future work we will evaluate the total cost (query + reply). Also, we plan to investigate the impact of non-TDMA MAC protocols on SRWs and NRWs.

Acknowledgement. This work has been funded by an IRCSET Postdoctoral Grant PD200857, SFI Grant No. SFI08-CE-I1380 and CONET, the Cooperating Objects Network of Excellence, EU FP7-2007-2-224053. The authors are thankful to Jan Hauer and Vlado Handziski for their support on the TWIST testbed.

References

1. http://www.graphviz.org
2. http://www.scilab.org
3. http://www.twist.tu-berlin.de/wiki
4. Ahn, J., Kapadia, S., Pattem, S., Sridharan, A., Zuniga, M., Jun, J., Avin, C., Krishnamachari, B.: Empirical evaluation of querying mechanisms for unstructured wireless sensor networks. SIGCOMM CCR 38(3), 17–26 (2008)
5. Avin, C., Brito, C.: Efficient and robust query processing in dynamic environments using random walk techniques. In: IPSN 2004 (2004)
6. Avin, C., Krishnamachari, B.: The power of choice in random walks: An empirical study. Computer Networks 52, 1 (2008)
7. Braginsky, D., Estrin, D.: Rumor routing algorthim for sensor networks. In: WSNA 2002 (2002)
8. Burioni, R., Cassi, D.: Random walks on graphs: ideas, techniques and results. J. Phys. A: Math. Gen. 38, R45–R78 (2005)
9. Cerpa, A., Wong, J.L., Kuang, L., Potkonjak, M., Estrin, D.: Statistical model of lossy links in wireless sensor networks. In: IPSN 2005 (2005)

10. Cerpa, A., Wong, J.L., Potkonjak, M., Estrin, D.: Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In: MobiHoc 2005. ACM, New York (2005)
11. Chang, N., Liu, M.: Revisiting the ttl-based controlled flooding search: optimality and randomization. In: MobiCom 2004. ACM, New York (2004)
12. Chang, N., Liu, M.: Controlled flooding search in a large network. IEEE/ACM Transactions on Networking (TON) 15(2), 449 (2007)
13. Cheng, Z., Heinzelman, W.: Flooding strategy for target discovery in wireless networks. Wireless Networks 11(5), 607–618 (2005)
14. Friedman, R., Kliot, G., Avin, C.: Probabilistic quorum systems in wireless ad hoc networks. In: IEEE DSN 2008, June 2008, pp. 277–286 (2008)
15. Lawler, G.F.: A self-avoiding random walk. Duke Math. J. 47(3), 655–693 (1980)
16. Lovasz, L.: Random walks on graphs: A survey. Combinatorics, Paul Erdos is Eighty 2(1), 1–46 (1993)
17. Ni, S., Tseng, Y., Chen, Y., Sheu, J.: The broadcast storm problem in a mobile ad hoc network. In: MOBICOM 1999, p. 162. ACM, New York (1999)
18. Pemantle, R.: Vertex-reinforced random walk. Probability Theory and Related Fields 92(1), 117–136 (1992)
19. Sadagopan, N., Krishnamachari, B., Helmy, A.: Active query forwarding in sensor networks. Ad Hoc Networks 3(1), 91–113 (2005)
20. Servetto, S., Barrenechea, G.: Constrained random walks on random graphs: Routing algorithms for large scale wireless sensor networks. In: WSNA 2002 (2002)
21. Shakkottai, S.: Asymptotics of query strategies over a sensor network. In: IEEE INFOCOM, Citeseer, vol. 1, pp. 548–557 (2004)
22. Simsek, O., Jensen, D.: Navigating networks by using homophily and degree. PNAS 105, 35 (2008)
23. Sohrabi, K., Manriquez, B., Pottie, G.: Near-ground wideband channel measurements. In: IEEE Proc. VTC, New York (1999)
24. Srinivasan, K., Kazandjieva, M., Agarwal, S., Levis, P.: The beta-factor: measuring wireless link burstiness. In: SenSys 2008 (2008)
25. Woo, A., Tong, T., Culler, D.: Taming the underlying challenges of reliable multihop routing in sensor networks. In: SenSys 2003 (2003)
26. Zeng, X., Bagrodia, R., Gerla, M.: GloMoSim: a library for parallel simulation of large-scale wireless networks. ACM SIGSIM Simulation Digest (1998)
27. Zhao, J., Govindan, R.: Understanding packet delivery performance in dense wireless sensor networks. In: SenSys 2003, New York, NY, USA (2003)
28. Zuniga, M., Krishnamachari, B.: An analysis of unreliability and asymmetry in low-power wireless links. ACM Trans. Sen. Netw. 3(2), 7 (2007)

TARF: A Trust-Aware Routing Framework for Wireless Sensor Networks*

Guoxing Zhan[1], Weisong Shi[1], and Julia Deng[2]

[1] Wayne State University, 5143 Cass Avenue, Detroit, MI 48202, USA
{gxzhan,weisong}@wayne.edu
[2] Intelligent Automation Inc., 15400 Calhoun, Rockville, MD 20855, USA
hdeng@i-a-i.com

Abstract. Multi-hop routing in wireless sensor networks (WSNs) offers little protection against deception through replaying routing information. This defect can be taken advantage of by an adversary to misdirect significant network traffic, resulting in disastrous consequences. It cannot be solved solely by encryption or authentication techniques. To secure multi-hop routing in WSNs against intruders exploiting the replay of routing information, we propose TARF, a trust-aware routing framework for WSNs. Not only does TARF significantly reduce negative impacts from these attackers, it is also energy-efficient with acceptable overhead. It incorporates the trustworthiness of nodes into routing decisions and allows a node to circumvent an adversary misdirecting considerable traffic with a forged identity attained through replaying. Both our empirical and simulated experimental results indicate that TARF satisfactorily performs routing and is resilient against attacks by exploiting the replay of routing information.

1 Introduction

Wireless sensor networks (WSNs) are ideal candidates for applications such as military surveillance and forest fire monitoring to report detected events of interest. With a narrow radio communication range, a sensor node wirelessly sends messages to a base station via a multi-hop path. However, the multi-hop routing of WSNs often becomes the target of malicious attacks. In such an attack, the attacker may tamper nodes physically, create traffic collision with seemingly valid transmission, drop or misdirect messages in routes, or jam the communication channel by creating radio interference [18]. This paper focuses on the kind of attack in which an adversary misdirects packets by identity deception through replaying routing information. With such identity deception, the adversary is capable of launching harmful and hard-to-detect attacks to misdirect traffic, such as selective forwarding as well as wormhole and sinkhole attacks [8].

As an effective and easy-to-implement type of attack, a malicious node simply replays all the routing information sent from another valid node to forge the latter node's identity, thus misdirecting the network traffic. Those packets, including their original headers, are replayed without any modification. Even if this malicious node cannot directly overhear the valid node's wireless transmission, it can collude with other malicious nodes to receive those routing packets and replay them somewhere far away from

* This work is supported in part by NSF grant CNS-0721456.

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 65–80, 2010.

the original valid node, which is known as a wormhole attack. Since a node in a WSN usually relies solely on the packets received to know about the sender's identity, replaying routing packets allows the malicious node to forge the identity of this valid node. After "stealing" that valid identity, this malicious node is able to misdirect the network traffic. In a selective forwarding attack, it may drop packets received, forward packets to another node not supposed to be in the routing path, or even form a transmission loop through which packets are passed among a few malicious nodes infinitely. It is often difficult to know whether a node forwards received packets correctly even with overhearing techniques [8]. Sinkhole attacks are another kind of attacks that can be launched after stealing a valid identity. In a sinkhole attack, a malicious node may claim itself to be a base station through replaying all the packets from a real base station. Such a fake base station could lure more than half the traffic, creating a "black hole".

Unfortunately, most existing routing protocols for WSNs either focus on energy efficiency [1] assuming that each node is honest with its identity, or they try to exclude unauthorized participation by encrypting data and authenticating packets. Examples of these encryption and authentication schemes for WSNs include TinySec [7], Spins [14], TinyPK [16], and TinyECC [10]. Admittedly, it is important to consider efficient energy usage for battery-powered sensor nodes and the robustness of routing under topological changes and common faults in a wild environment. However, it is also significant to incorporate security as one of the most important goals; meanwhile, even with perfect encryption and authentication, by replaying routing information, a malicious node can still participate in the network using another valid node's identity.

In contrast, trust management [2] has been introduced into peer-to-peer networks and general ad hoc networks to support decision-making [6,15], improve security [3,11], and promote node collaboration [5] and resource sharing [9]. Basically, trust management assigns each node a trust value according to its past performance. These studies target general ad hoc networks and peer-to-peer networks but not resource-constrained WSNs. Additionally, they do not address attacks arising from the replay of routing information. With a similar idea, S. Ganeriwal, L. Balzano, and M. Srivastava also proposed a reputation-based approach to detect uncooperative nodes in WSNs [4]; however, they do not address the attacks by exploiting the replay of routing information. The authors also studied the trustworthiness of the data collected by WSNs [19].

At this point, to fight against the "identity theft" threat arising from packet replaying, we introduce trust management into WSNs, proposing TARF - a trust-aware routing framework for wireless sensor networks. TARF identifies those malicious nodes that misuse "stolen" identities to misdirect packets by their low trustworthiness, thus helping nodes circumvent those attackers in their routing paths. We present the assumptions and goals of this work in Section 2, the detailed design of TARF in Section 3, our implementation of TARF in Section 4 and simulation results in Section 5. Finally, we conclude this work in Section 6.

2 Assumptions and Goals

We target secure routing for data collection tasks, which are one of the most fundamental functions of WSNs. In a data collection task, sensor nodes send sampled data to a

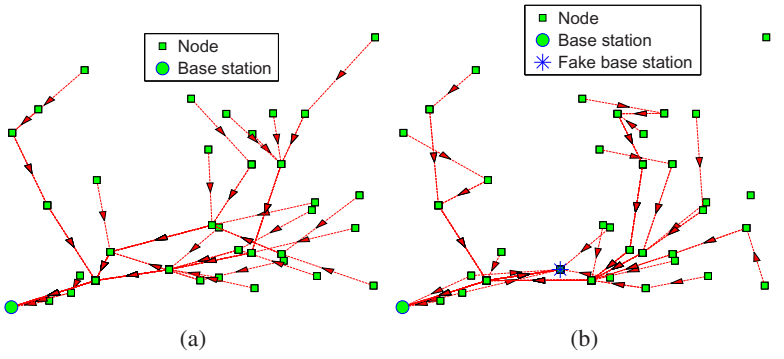


Fig. 1. Multi-hop routing: (a) normal scenarios; (b) a fake base station attracts traffic

remote base station with the aid of intermediate nodes, as in Figure 1(a). It is possible for an adversary to replay all the packets from a base station and thus to forge the identity of the base station. Such deception could result in the following situation: a large amount of packets are attracted to this fake base station and are never delivered to the real base station (see Figure 1(b)).

Though there could be more than one base station, our routing approach is not affected by the number of base stations; to simplify our discussion, we will assume that there is only one base station. Further, we assume no data aggregation is involved. Nonetheless, our approach can still be applied to static-cluster-based WSNs, where data are aggregated by static clusters before being relayed. In a static-cluster-based WSN, cluster headers themselves form a sub-network; after certain data reach a cluster header, the aggregated data will be routed to a base station only through such a sub-network consisting of cluster headers. Our framework can then be applied to this sub-network to achieve secure routing for static-cluster-based WSNs.

Additionally, we make certain assumptions regarding the format of packets in TARF. We assume all data packets and routing packets, including their packet headers, are authenticated; a packet can be forwarded only after its authenticity is verified. Whether data encryption is implemented can be decided by the application. Every data packet is assumed to have at least the following fields: the sender id, the sender sequence number, the next-hop node id (the receiver in this one-hop transmission), the source id (the node that initiates the data), and the source's sequence number. We insist that the source node's information should be included for the following reasons. First, that allows the base station to identify which data packets are initiated but undelivered; Second, a WSN cannot afford the overhead to transmit all the one-hop information to the base station. Regarding routing packets, they should have at least the following fields: the source id, the source's sequence number, and the next-hop id. In addition, we assume that after receiving a data packet, a node will send out an acknowledgement packet.

Next, we present the goals of TARF.

High Throughput: *Throughput* is defined as the ratio of the number of data packets delivered to the base station to the number of all sampled data packets. Note that single-hop re-transmission may happen, and that identical packets repeatedly transmitted are

considered as one packet as far as *throughput* is concerned. Instead of any specific data, users usually care much more about *throughput*. Here we regard high *throughput* as one of our most important goals.

Energy Efficiency: Efficient energy usage is significant for battery-powered sensor nodes, and data transmission accounts for a major portion of energy consumption. We evaluate energy efficiency by the average energy cost to successfully deliver a unit-sized data packet from a source node to the base station. Note that link-level re-transmission should be given enough attention when considering energy cost since each re-transmission causes a noticeable increase in energy consumption. If every node in a WSN consumes approximately the same energy to transmit a unit-sized data packet, we can use another metric *hop-per-delivery* to evaluate energy efficiency. Under that assumption, the energy consumption depends on the number of hops, i.e. the number of one-hop transmissions occurring. To evaluate how efficiently energy is used, we can measure the average hops per delivery, i.e., the number of all hops divided by the number of all delivered data packets, abbreviated as *hop-per-delivery*.

Excellent Scalability & Adaptability: TARF should work well with WSNs of large magnitude under highly dynamic contexts.

Here we do not include other aspects such as latency, load balance, or fairness. Low latency, balanced network load, and good fairness requirements can be enforced in specific routing protocols built on top of TARF.

3 Design of TARF

TARF secures the multi-hop routing in WSNs against intruders exploiting the replay of routing information by evaluating the trustworthiness of neighboring nodes. It identifies such intruders that misdirect noticeable network traffic by their low trustworthiness and routes data through paths circumventing those intruders to achieve satisfactory *throughput*. TARF is also energy-efficient, highly scalable, and well adaptable. Before introducing the detailed design, we first introduce several necessary notions here.

Neighbor: For a node N, a neighbor (neighboring node) of N is a node that is reachable from N with one-hop wireless transmission.

Trust level: For a node N, the trust level of a neighbor is a decimal number in $[0, 1]$, representing N's opinion of that neighbor's level of trustworthiness. Specifically, the trust level of the neighbor is N's estimation of the probability that this neighbor correctly delivers data received to the base station. That trust level is denoted as T in this paper.

Energy cost: For a node N, the energy cost of a neighbor is the average energy cost to successfully deliver a unit-sized data packet with this neighbor as its next-hop node, from N to the base station. That energy cost is denoted as E in this paper.

3.1 Overview

TARF integrates trustworthiness and energy efficiency in making routing decisions. For a node N to route a data packet to the base station, N only needs to decide to which

neighboring node it should forward the data packet. That chosen neighbor is N's next-hop node. Once the data packet is forwarded to that next-hop node, the remaining task to deliver the data to the base station is fully delegated to it, and N is totally unaware of what routing decision its next-hop node makes. To choose its next-hop node, N considers both the trustworthiness and the energy efficiency of its neighbors. For that, N maintains a neighborhood table with trust level values and energy cost values for certain known neighbors. It is sometimes necessary to delete some neighbors' entries to keep the table size acceptable. Maintaining a neighborhood table with acceptable overhead proved possible in [17]; the same technique can be used by TARF.

In TARF, in addition to data packet transmission, there are two types of routing information that need to be exchanged: broadcast messages from the base station about undelivered data packets and energy cost report messages from each node. Neither message needs acknowledgement. A broadcast message from the base station is broadcast to the whole network; each node receiving a fresh broadcast message from the base station will broadcast it to all its neighbors once. The freshness of a broadcast message is ensured by its field of source sequence number. The other type of exchanged routing information is the energy cost report message from each node, which is broadcast to only its neighbors once. Additionally, any node receiving such an energy cost report message will not forward it.

For each node N in a WSN, to maintain such a neighborhood table with trust level values and energy cost values for certain known neighbors, two components, *EnergyWatcher* and *TrustManager*, run on the node (Figure 2). *EnergyWatcher* is responsible for recording the energy cost for each known neighbor, based on N's observation of one-hop transmission to reach its neighbors and the energy cost report from those neighbors. *TrustManager* is responsible for tracking trust level values of neighbors based on network loop discovery and broadcast messages from the base station about undelivered data packets. Once N is able to decide its next-hop neighbor according to its neighborhood table, it sends out its energy report message: it broadcasts to all its neighbors its energy cost to deliver a packet from the node to the base station. The energy cost is computed as in Section 3.3 by *EnergyWatcher*. Such an energy cost report also serves as the input of its receivers' *EnergyWatcher*.

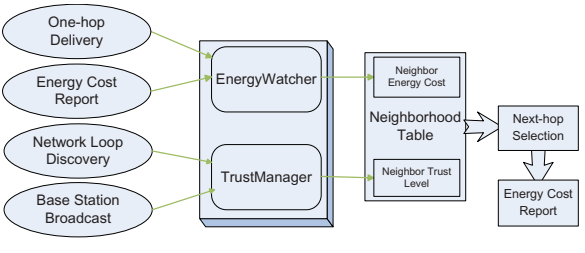


Fig. 2. Each node selects a next-hop node based on its neighborhood table, and broadcast its energy cost within its neighborhood. To maintain this neighborhood table, *EnergyWatcher* and *TrustManager* on the node keep track of related events (on the left) to record the energy cost and the trust level values of its neighbors.

3.2 Routing Procedure

TARF, as with many other routing protocols, runs as a periodic service. The length of that period determines how frequently routing information is exchanged and updated. At the beginning of each period, the base station broadcasts the information about undelivered data packets during the past few periods to the whole network once, which triggers the exchange of routing information in this new period. Whenever a node receives such a broadcast message from the base station, it knows that the most recent period has ended and a new period has just started. In this way, no time synchronization is required for a node to keep track of the beginning or ending of a period. During each period, the *EnergyWatcher* on a node monitors energy consumption of one-hop transmission to its neighbors and processes energy cost reports from those neighbors to maintain energy cost entries in its neighborhood table; its *TrustManager* also keeps track of network loops and processes broadcast messages from the base station about undelivered data to maintain trust level entries in its neighborhood table.

To maintain the stability of its routing path, a node may retain the same next-hop node until the next fresh broadcast message from the base station occurs. Meanwhile, to reduce traffic, its energy cost report could be configured to not occur again until the next fresh broadcast from the base station. If a node does not change its next-hop node selection until the next broadcast from the base station, that guarantees all paths to be loop-free, as can be deducted from the procedure of next-hop node selection. However, as noted in our experiments, that would lead to slow improvement in routing paths. Therefore, we allow a node to change its next-hop selection in a period only when its current next-hop is not responding correctly.

Next, we introduce the structure and exchange of routing information as well as how nodes make routing decisions in TARF.

Structure and Exchange of Routing Information: A broadcast message from the base station fits into a fixed number of packets; in our implementation, it fits into one byte. Such a message consists of a few pairs of <the node id of a source node, an undelivered sequence interval [a, b] with a significant length>. To reduce overhead, only a few such pairs are selected to be broadcast. The undelivered sequence interval [a, b] is explained as follows: the base station searches the source sequence numbers received in the past few periods, identifies which source sequence numbers for the source node with this id are missing, and chooses certain significant interval [a, b] of missing source sequence numbers as an undelivered sequence interval. For example, the base station may have all the source sequence numbers for the source node 2 as {109, 110, 111, 150,151} in the past two periods. Then [112, 149] is an undelivered sequence interval. Since the base station is usually connected to a powerful platform such as a desktop, a program can be developed on that powerful platform to assist in recording all the source sequence numbers and finding undelivered sequence intervals. The reason for searching over more than one period is to identify as many undelivered data packets as possible. To illustrate that, consider this example: suppose the source sequence numbers of delivered data packets from node 2 are {1, 2, 3} for the 1st period and {200, 201, 203} for the 2nd period; then simply searching over a single period would not discover the undelivered packets unless every node is required to send a fixed number of data packets over each period.

Accordingly, each node in the network stores a table of <the node id of a source node, a forwarded sequence interval [a, b] with a significant length> in the past few periods. The data packets with the source node and the sequence numbers falling in this forwarded sequence interval [a, b] have already been forwarded by this node. When the node receives a broadcast message with undelivered sequence intervals, its *TrustManager* will be able to identify which data packets forwarded by this node are not delivered to the base station. Considering the overhead to store such a table, old entries will be deleted once the table is full.

Once a fresh broadcast message from the base station is received, a node immediately invalidates all the existing energy cost entries: it is ready to receive a new energy report from its neighbors and choose its new next-hop node afterwards. Also, it is going to select a node either after a timeout is reached or after it has received an energy cost report from some highly trusted candidates with acceptable energy cost. A node immediately broadcasts its energy cost to its neighbors only after it has selected a new next-hop node. That energy cost is computed by its *EnergyWatcher* (see Section 3.3). A natural question is which node starts reporting its energy cost first. For that, note that when the base station is sending a broadcast message, a side effect is that its neighbors receiving that message will also regard this as an energy report: the base station needs 0 amount of energy to reach itself. As long as the original base station is faithful, it will be viewed as a trustworthy candidate by *TrustManager* on the neighbors of the base station. Therefore, those neighbors will be the first nodes to decide their next-hop node, which is the base station; they will start reporting their energy cost once that decision is made.

Route Selection: Now, we introduce how TARF decides routes in a WSN. Each node N relies on its neighborhood table to select an optimal route, considering both energy consumption and reliability. TARF makes good efforts in excluding those nodes that misdirect traffic by exploiting the replay of routing information.

For a node N to select a route for delivering data to the base station, N will select an optimal next-hop node from its neighbors based on trust level and energy cost and forward the data to the chosen next-hop node immediately. The neighbors with trust levels below a certain threshold will be excluded from being considered as candidates. Among the remaining known neighbors, N will select as its next-hop node a neighbor b with the minimal value of $\frac{E_{Nb}}{T_{Nb}}$, with E_{Nb} and T_{Nb} being b's energy cost and trust level value in the neighborhood table respectively (see Section 3.3, 3.4). Basically, E_{Nb} reflects the energy cost of delivering a packet to the base station from N assuming that all the nodes in the route are honest; $\frac{1}{T_{Nb}}$ approximately reflects the number of the needed attempts to send a packet from N to the base station via multiple hops before such an attempt succeeds, considering the trust level of b. Thus, comparing the values of $\frac{E_{Nb}}{T_{Nb}}$ among N's neighbors identifies a candidate with a minimal combined cost of energy and trustworthiness.

The remaining delivery task is fully delegated to that selected next-hop neighbor, and N is totally unaware of what routing decision its chosen neighbor is going to make. Next, the chosen node will repeat what N has done, i.e., delegating the left routing task to its own chosen next-hop neighbor. In this way, instead of finding out a complete path to the base station, each node is only responsible for choosing its next-hop node,

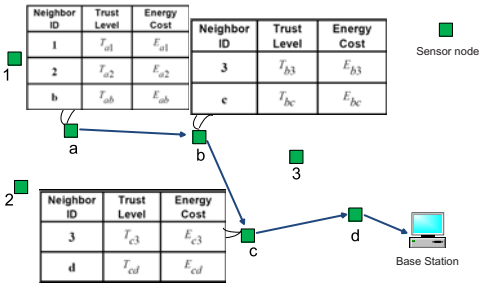


Fig. 3. Routing illustration

thus saving considerable cost in computation and routing information exchange. As an example shown in Figure 3, node a is trying to forward a packet to the base station. After comparing both the trust level and energy cost among its neighbors 1, 2 and b, a decides that b is the most promising next-hop node for data delivery and forwards the data packet to b immediately. b is free to make its own decision for routing the packet to the base station. b decides that its neighbor c is a better candidate than its neighbor 3. After that, the task is delegated to c, and c continues to delegate the job to d. Finally, d delivers the packet to the base station. Observe that in an ideal misbehavior-free environment, all nodes are absolutely faithful, and each node will choose a neighbor through which the routing path is optimized in terms of energy; thus, an energy-driven route is achieved. If we further assume that the one-hop transmission power of a unit-sized packet is the same for each node, the selected route will be the classical shortest path.

3.3 EnergyWatcher

Here we describe how a node N's *EnergyWatcher* computes the energy cost E_{Nb} for its neighbor b in N's neighborhood table and how N decides its own energy cost E_N. Before going further, we will clarify some notations. E_{Nb} mentioned is the average energy cost of successfully delivering a unit-sized data packet from N to the base station, with b as N's next-hop node being responsible for the remaining route. Here, one-hop re-transmission may occur until the acknowledgement is received or the number of re-transmissions reaches a certain threshold. The cost caused by one-hop re-transmissions should be included when computing E_{Nb}. Suppose N decides that A should be its next-hop node after comparing energy cost and trust level. Then N's energy cost is $E_N = E_{NA}$. Denote $E_{N\to b}$ as the average energy cost of successfully delivering a data packet from N to its neighbor b with one hop. Note that the re-transmission cost needs to be considered. With the above notations, it is straightforward to establish the following relation:

$$E_{Nb} = E_{N\to b} + E_b$$

Since each known neighbor b of N is supposed to broadcast its own energy cost E_b to N, to compute E_{Nb}, N still needs to know the value $E_{N\to b}$, i.e., the average energy cost of successfully delivering a data packet from N to its neighbor b with one hop. For that,

assuming that the endings (being acknowledged or not) of one-hop transmissions from N to b are independent with the same probability p_{succ} of being acknowledged, we first compute the average number of one-hop sendings needed before the acknowledgement is received as follows:

$$\sum_{i=1}^{\infty} i \cdot p_{succ} \cdot (1 - p_{succ})^{i-1} = \frac{1}{p_{succ}}$$

Denote E_{unit} as the energy cost for node N to send a unit-sized data packet once regardless of whether it is received or not. Then we have

$$E_{Nb} = \frac{E_{unit}}{p_{succ}} + E_b$$

The remaining job for computing E_{Nb} is to get the probability p_{succ} that a one-hop transmission is acknowledged. Considering the variable wireless connection among wireless sensor nodes, we do not use the simplistic averaging method to compute p_{succ}. Instead, after each transmission from N to b, N's *EnergyWatcher* will update p_{succ} based on whether that transmission is acknowledged or not with a weighted averaging technique. We use a binary variable Ack to record the result of current transmission: 1 if an acknowledgement is received; otherwise, 0. Given Ack and the last probability value of an acknowledged transmission p_{old_succ}, TARF uses a weighted average of Ack and p_{old_succ} as the new probability value p_{new_succ}:

$$p_{new\_succ} = (1 - w) \times p_{old\_succ} + w \times Ack, w \in (0, 1),$$

where w can be chosen by specific protocols.

3.4 TrustManager

A node N's *TrustManager* decides the trust level of each neighbor based on the following events: discovery of network loops, and broadcast from the base station about undelivered data packets. For each neighbor b of N, T_{Nb} denotes the trust level of b in N's neighborhood table. At the beginning, each neighbor is given a neutral trust level 0.5. After any of those events occurs, the relevant neighbors' trust levels are updated.

To detect loops, the *TrustManager* on N reuses the table of <the node id of a source node, a forwarded sequence interval [a, b] with a significant length> (see Section 3.2) in the past few periods. If N finds that a received data packet is already in that record table, not only will the packet be discarded, but the TrustManager on N also degrades its next-hop node's trust level. If that next-hop node is b, then T_{old_Nb} is the latest trust level value of b. We use a binary variable $Loop$ to record the result of loop discovery: 1 if a loop is received; 0 otherwise. After the degradation, as in the update of energy cost, the new trust level of b is

$$T_{new\_Nb} = (1 - w) \times T_{old\_Nb} + w \times Loop, w \in (0, 1),$$

where w can be chosen by specific applications.

Once a loop has been detected by N for a few times so that the trust level of the next-hop node is too low, N will change its next-hop selection; thus, that loop is broken. Though N can not tell which node should be held responsible for the occurrence of a loop, degrading its next-hop node's trust level gradually leads to the breaking of the loop.

On the other hand, to detect the traffic misdirection by nodes exploiting the replay of routing information, *TrustManager* on N compares N's stored table of <node id of a source node, forwarded sequence interval [a, b] with a significant length> recorded in the past few periods with the broadcast messages from the base station about undelivered data. It computes the ratio of the number of successfully delivered packets which are forwarded by this node to the number of those forwarded data packets, denoted as $DeliveryRatio$. Then N's *TrustManager* updates its next-hop node b's trust level as follows:

$$T_{new\_Nb} = (1 - w) \times T_{old\_Nb} + w \times DeliveryRatio, w \in (0, 1),$$

Now, suppose an adversary M forges the identity of the base station by replaying all the routing packets from the base station. At first, it is able to deceive its neighbors into believing that M is a base station; as a result, M may attract a large amount of data packets, which never reach the base station. However, after the base station broadcasts the information about those undelivered packets, M's neighbors will downgrade M's trust level values in their neighborhood table. Note that M is only capable of replaying but is not capable of manipulating or generating authenticated broadcast messages, and that M usually cannot prevent other nodes from receiving a broadcast message from the base station. As time elapses, M's neighbors will start realizing that M is not trustworthy and will look for other next-hop candidates that are more reliable. Similarly, if M forges the identity of another valid appealing node, M's neighbors will gradually realize that M is not reliable.

4 Implementation and Empirical Evaluation

We have implemented a protocol based on TARF in TinyOS 1.x, which currently runs on mica2 motes. Both the authentication and encryption of packets reuse the implementation of TinySec [7]: TinySec uses a CBC mode encryption scheme with Skipjack as the block cipher and an authentication scheme based on a four-byte message authentication code (MAC) computed by the CBC-MAC construction procedure. The MAC field is computed over the whole message including all the headers; it also serves as the CRC field of the packet. Data encryption can be disabled. In a routing packet, the next-hop id is replaced by a neighborhood broadcast address or a network broadcast address to indicate that it is a neighborhood or whole network broadcast. The acknowledgement of data packets is enabled. Considering the fact that floating-point computation is not supported by sensor hardware, the implementation uses an integer in $[0, 100]$ to represent trust level; the update of energy cost and trust level values is also implemented using integer arithmetics.

This implemented TARF protocol requires moderate program storage and memory usage. For comparison, we list the ROM size and RAM size requirement for this protocol and two other protocols on mica nodes in Table 1. The two other protocols are

Table 1. Size of protocol components implemented

Protocol	Authentication&Encryption	ROM (bytes)	RAM (bytes)
TARF	TinySec	20912	1464
Route	TinySec	20696	1048
MintRoute	TinySec	22554	1990

named Route and MintRoute according to their directory name under TinyOS 1.x. Both Route and MintRoute were the "standard" routing protocols in TinyOS 1.x and make route decisions based on both link quality estimation and number of hops. Neither of these original protocols provides encryption or authentication; to compare on a fair basis, we also enabled the encryption and authentication mode of TinySec for Route and MintRoute. TinySec occupies 728 bytes of RAM and 7146 bytes of ROM [7]. Similarly to Route and MintRoute, this TARF protocol adopts energy-efficient routes in a misbehavior-free environment. However, with a comparable size, it also supports the circumvention of adversaries exploiting the replay of routing information, which is not provided by Route or MintRoute. Further, our experience shows that it is easy to incorporate this TARF protocol into most applications. As an example, we re-implemented the Surge application in the TinyOS 1.x directory with this TARF protocol. The program has a size comparable to that of the Surge implemented using Route or MintRoute.

To evaluate how effective TARF is against deception through replaying routing information in the real world, we uploaded programs onto Motelab [13] at Harvard University. As a public test bed of wireless sensor networks, at the time of our experiments, 184 TMote Sky sensor motes were deployed at 3 floors. These nodes are distributed among many rooms of the building, with an approximate indoor transmission of 100 meters. Approximately 14 nodes were removed, and nearly 50 nodes were disabled. Motelab switched its serial forwarder protocol from TinyOS 1.x to TinyOS 2.x and was equipped with TMote only Tmote Sky motes. Due to the unavailability of TinySec on TMote SKy nodes, we did not include authentication or encryption from TinySec in the uploaded programs. Further, considering the availability of routing protocols on TinyOS 2.x, we compared our TinyOS 2.x version of TARF with the collection tree routing protocol (CTP), which mainly employs link quality estimation in choosing next-hop nodes. Both protocols were integrated into a data collection application - MultihopOscilloscope, which is named after its directory name in TinyOS 2.x. We configured the MultihopOscilloscope to send out 5 samples in a single data packet every 5 seconds. The routing update occurred every 50 seconds. Because of the limited quota assigned by Motelab, our programs lasted maximally 30 minutes. Among all the nodes, one was chosen to be the base station. Another node was programmed to be a fake base station: it broadcast as if it were a base station but never delivered the received data to the real base station. The many experiments we executed indicate that our TARF protocol achieves at least 30% higher *throughput* than CPT when there is an "attractive" fake base station. Some fake base stations are not able to misdirect much traffic because they have a poor wireless connection with their neighbors and do not look "appealing".

In one experiment (Figure 4(a)), all nodes on the three floors were supposed to deliver data to node 9 (the base station); node 15 (fake base station) replayed all the routing packets from the base station. By counting the data packets received at the real base station, TARF had approximately a 60% higher *throughput* than CTP. In another experiment (Figure 4(b)), only the nodes on the first floor (56 nodes totally) sent data to node 9 (the base station), and node 27 (fake base station) replayed the routing packets from the base station. As a result, TARF had approximately a 40% higher *throughput* than CTP.

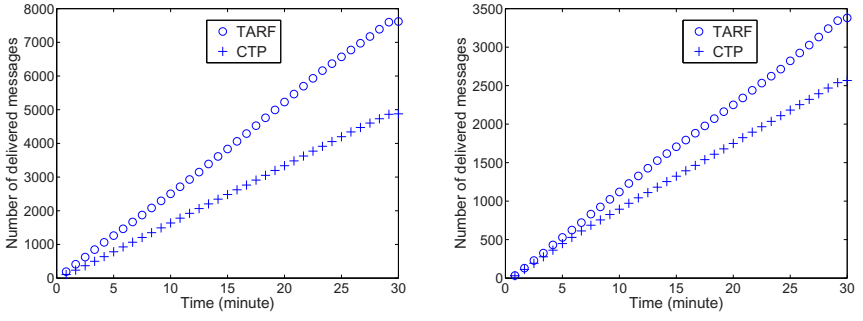


Fig. 4. With a fake base at Motelab, (a) TARF had approximately a 60% higher *throughput* than CTP among 3 floors; (b) TARF had approximately a 40% higher *throughput* than CTP at a single floor.

We also recorded the number of redundant data packets received by the base station. It turns out that both TARF and CTP had redundancy ratios at no more than 2%. Though both CTP and TARF suppress redundant packets, a packet might be received more than once by the base station because an acknowledgment is lost when the route changes.

5 Simulation and Evaluation

To further evaluate the efficacy of TARF in terms of energy efficiency and *throughput*, we have developed a reconfigurable emulator of wireless sensor networks on a two-dimensional plane with Matlab [12]. To effectively simulate a WSN, this emulator uses the object-oriented technique to construct two classes of objects: WSNMANAGER and NODE, to represent the whole network and a sensor node. The interaction between nodes are emulated through event passing. The routing function for a node can be rewritten to adopt different routing protocols; different maps can also be ported into this simulator. To simulate the unreliable wireless transmission, the outcome of one-hop packet transmission is decided by the following model: suppose a node A is wirelessly transmitting a packet to node B, the probability for B to successfully receive such a packet is assumed to be

$$1 - (min(dist, MAX\_DIST)/MAX\_DIST)^8,$$

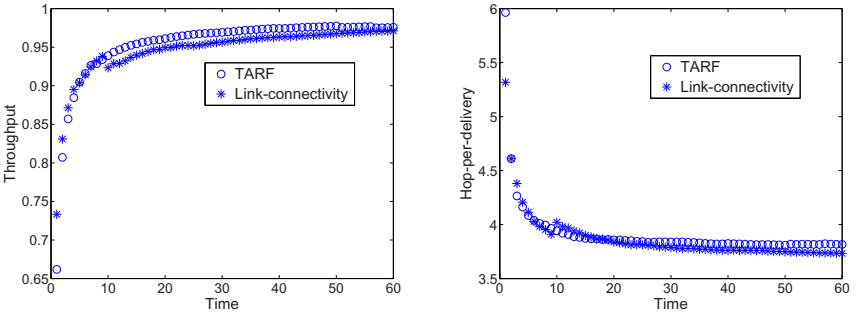
where $dist$ is the distance from A to B, and MAX_DIST is the maximal transmission range. In our experiment, MAX_DIST is defined as 100m, and 35 nodes are randomly distributed within a 300*300 rectangular area. All the nodes have the same power level and the same maximal transmission range of 100m. A base station is placed at the origin [0, 0]. We simulate the sensor network in 60 consecutive periods; each node samples data 6 times in each period.

The performance of TARF is compared to that proposed in [17] by Alec Woo, Terence Tong and David Culler. In that project, link connectivity is used as a cost metric for routing, which is found to be more cost-effective than the well-known shortest path protocol. We will simply refer to the latter protocol as link-connectivity. In our simulation experiments, we compare TARF with a simulated version of link-connectivity. As we will see from the experiment results, with the existence of misbehaviors, the *throughput* in TARF is often much higher than that in link-connectivity; the *hop-per-delivery* in TARF is generally at least comparable to that in the link connectivity protocol.

We compare TARF and link-connectivity under the following scenarios: (1) no nodes misbehaves intentionally; (2) certain nodes forge the identity of the based station by replaying broadcast messages; (3) a set of nodes colludes to form a forwarding loop; and (4) a set of nodes drops received data packets.

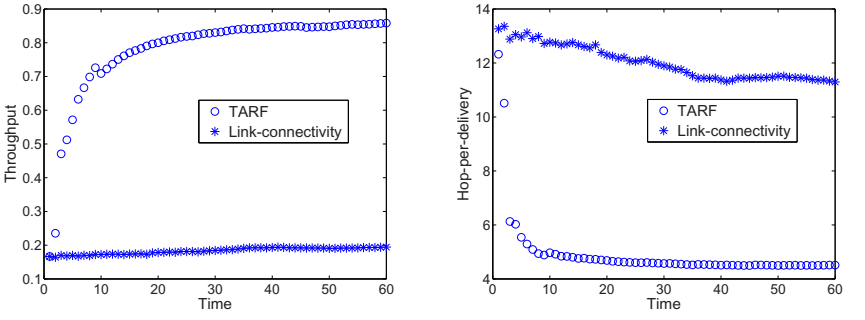
Under scenario (1) without misbehaving nodes, the two protocols have comparable performance in terms of *throughput* and *hop-per-delivery*. Figure 5 shows such an example. Under a misbehavior-free environment, according to the TARF protocol, a node may still perceive its neighbors as having different trust level, due to the fact that the node can not well distinguish between malicious behavior and failed delivery due to environmental effects. However, such mis-perception of trust does not compromise the performance of TARF.

Under scenario (2), certain malicious nodes become fake base stations through replaying messages originated from the base station. With the link connectivity protocol, a significant portion of traffic is attracted to the fake base. However, with TARF, most packets are able to select a route circumventing those fake bases. When there are forged base stations, TARF tends to show much better *throughput* than the link



(a) (b)

Fig. 5. Under misbehavior-free environment, TARF and link-connectivity have comparable performance in (a) *throughput*, and (b) *hop-per-delivery*.



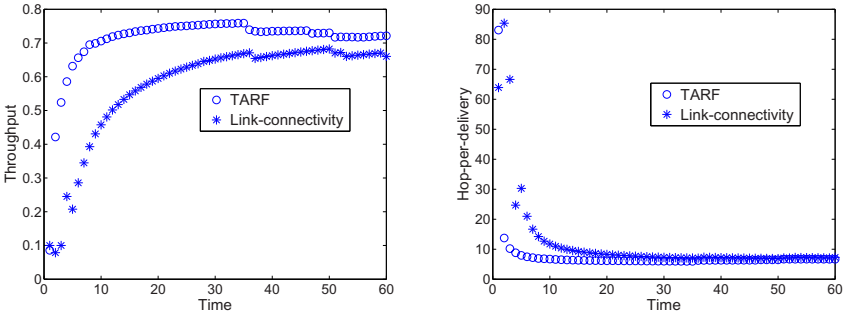
(a) (b)

Fig. 6. With a fake base, (a) TARF has 5 times the *throughput* in link-connectivity; (b) TARF has less than 50% *hop-per-delivery* in link-connectivity.

connectivity protocol, and the *hop-per-delivery* in TARF is much less than that in the link-connectivity protocol. In one of our experiments with a fake base station, as indicated in Figure 6, TARF reaches roughly 5 times the *throughput* in the link-connectivity protocol, while the *hop-per-delivery* in TARF is less than 50% that in link-connectivity.

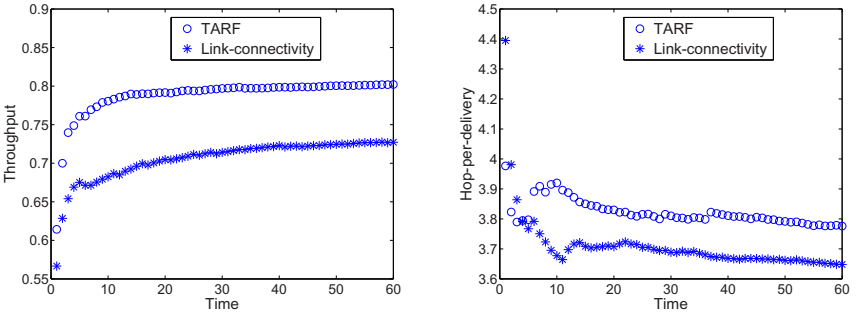
Under scenario (3), a loop of colluding nodes intercepts many packets. The *throughput* in TARF is generally higher than that in link-connectivity; the *hop-per-delivery* in the two protocols gradually become comparable. In one experiment, as shown in Figure 7, 5 out of 35 nodes are selected to form a network loop. Any data forwarded to one of these 6 nodes would not be able to arrive at the base station. As in Figure 7, the *throughput* in TARF is around 70% higher than that in the link connectivity protocol; their *hop-per-delivery* gradually becomes comparable.

Under scenario (4), a set of nodes drops any received data packets. In our experiment, 6 nodes drop data forwarded to them. As indicated by Figure 8, the *throughput* in TARF



(a) (b)

Fig. 7. With a loop consisting of 14% nodes, (a) TARF has a higher *throughput* than link-connectivity; (b) gradually, TARF and link-connectivity have comparable *hop-per-delivery*.



(a) (b)

Fig. 8. With 6 nodes dropping data, (a) TARF has a 14% higher *throughput* than link-connectivity; (b) TARF has a 5% higher *hop-per-delivery* than link-connectivity.

is at least 14% greater than that in link-connectivity; the *hop-per-delivery* in TARF is around 5% higher than that in link-connectivity.

6 Conclusions

We propose TARF, a trust-aware routing framework for WSNs, to secure multi-hop routing in WSNs against intruders exploiting the replay of routing information. With the idea of trust management, TARF enables a node to keep track of the trustworthiness of its neighbors and thus to select a reliable route. Not only does TARF circumvent those malicious nodes misusing other nodes' identities to misdirect network traffic, it also accomplishes efficient energy usage. Our implementation and simulation results indicate that (1) the efficiency of energy usage in TARF is generally at least comparable to that in existing protocols; (2) with the existence of traffic misdirection through "identity theft", TARF generally achieves a significantly higher *throughput* than other existing protocols; and (3) TARF is scalable and adaptable to typical medium-scale testbed environments and simulated conditions. Our future work is to further evaluate TARF with large-scale WSNs deployed in wild environments and to study how to choose parameters involved for specific applications. We believe that the idea of TARF can also be applied to general ad hoc networks and peer-to-peer networks to fight against similar attacks.

References

1. Al-Karaki, J., Kamal, A.: Routing techniques in wireless sensor networks: a survey. IEEE Wireless Communications 11(6), 6–28 (2004)
2. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In: Proceedings of 1996 IEEE Symposium on Security and Privacy, pp. 164–173 (1996)
3. 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 (2004)

4. Ganeriwal, S., Balzano, L., Srivastava, M.: Reputation-based framework for high integrity sensor networks. ACM Trans. Sen. Netw. (2008)
5. He, Q., Wu, D., Khosla, P.: Sori: A secure and objective reputation-based incentive scheme for ad hoc networks. In: Proceedings of IEEE Wireless Communications and Networking Conference, pp. 825–830 (2004)
6. Kamvar, S., Schlosser, M., Garcia-Molina, H.: The eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the 12th international conference on World Wide Web, pp. 640–651 (2003)
7. Karlof, C., Sastry, N., Wagner, D.: Tinysec: A link layer security architecture for wireless sensor networks. In: Proc. of ACM SenSys 2004 (November 2004)
8. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: attacks and countermeasures. In: First IEEE International Workshop on Sensor Network Protocols and Applications (2003)
9. Liang, Z., Shi, W.: Pet: A personalized trust model with reputation and risk evaluation for p2p resource sharing. In: HICSS 2005: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS 2005) - Track 7. IEEE Computer Society, Los Alamitos (2005)
10. Liu, A., Ning, P.: Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In: IPSN 2008: Proceedings of the 7th international conference on Information processing in sensor networks, pp. 245–256. IEEE Computer Society, Los Alamitos (2008)
11. Liu, Z., Joy, A., Thompson, R.: A dynamic trust model for mobile ad hoc networks. In: Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, pp. 80–85 (2004)
12. Matlab, http://www.mathworks.com
13. Motelab, http://motelab.eecs.harvard.edu
14. Perrig, A., Szewczyk, R., Wen, W., Culler, D., Tygar, J.: SPINS: Security protocols for sensor networks. Wireless Networks Journal (WINET) 8(5), 521–534 (2002)
15. Wang, Y., Vassileva, J.: Trust and reputation model in peer-to-peer networks. In: Proceedings of the 3rd International Conference on Peer-to-Peer Computing, p. 150 (2003)
16. Watro, R., Kong, D., Cuti, S., Gardiner, C., Lynn, C., Kruus, P.: Tinypk: securing sensor networks with public key technology. In: SASN 2004: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, pp. 59–64. ACM, New York (2004)
17. Woo, A., Tong, T., Culler, D.: Taming the underlying challenges of reliable multihop routing in sensor networks. In: Proceedings of the First ACM SenSys 2003 (November 2003)
18. Wood, A., Stankovic, J.: Denial of service in sensor networks. Computer 35(10), 54–62 (2002)
19. Zhan, G., Shi, W., Deng, J.: Poster abstract: Sensortrust - a resilient trust model for wsns. In: SenSys 2009: Proceedings of the 7th International Conference on Embedded Networked Sensor Systems (2009)

Low-Overhead Dynamic Multi-channel MAC for Wireless Sensor Networks

Joris Borms[1], Kris Steenhaut[1,2], and Bart Lemmens[1,2]

[1] Vrije Universiteit Brussel, Dept. of Electronics and Informatics ETRO
[2] Erasmus Hogeschool Brussel, Dept. of Industrial Sciences and Technology
Brussels, Belgium
{jborms,ksteenha,blemmens}@etro.vub.ac.be

Abstract. Most of the existing popular MAC protocols for Wireless Sensor Networks (WSN) only use a single channel for relaying data. Most popular platforms however are equipped with a radio chip capable of switching its channel, and are therefor not restricted to a single-channel operation. Operating on multiple channels can increase bandwidth and can provide robustness against external interference. We argue that this feature is not only useful for dense, high-throughput WSNs but also for sparser networks with low average data rates but with occasional traffic bursts. We present MuChMAC, a low-overhead **Mu**lti-**Ch**annel MAC protocol which uses a combination of TDMA and asynchronous MAC techniques to exploit multi-channel operation without the need for coordination or tight synchronization between nodes. We describe an interface to scale MuChMAC's duty cycle to adapt to varying traffic conditions or energy constraints. We demonstrate MuChMAC's usefulness on a testbed consisting out Sentilla JCreate motes running it as the MAC layer for Contiki-based applications.

1 Introduction

Traditional solutions for Medium Access Control (MAC) in Wireless Sensor Networks (WSN) use only a single frequency channel for sending and receiving messages. This implies that the bandwidth of a single channel has to be shared amongst all nodes in the same neighborhood. Efficiently sharing bandwidth while keeping low power consumption is a challenging task. Traditional MAC algorithms for WSNs can be divided in different categories depending on how they try to tackle these problems. Some algorithms try to synchronize all nodes' duty cycles and then schedule transmissions according to either contention-based (e.g. TMAC [1]) or TDMA-based methods (e.g. LMAC [2]) or a mix of both. Others avoid synchronization and focus more on keeping nodes in a low duty cycle (e.g. B-MAC [3], X-MAC [4]) reasoning that the chances of collision are small since one expects a small amount of traffic on the network. As such the overhead of resolving collisions is also small. This last category is a better fit for sensor networks with very low duty cycles since no coordination or synchronization between the nodes is required, but they do run into throughput

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 81–96, 2010.

problems when the requested data rates of nodes increase. Additionally, any single-channel MAC protocol can run into problems when its channel is being used by external devices.

In the rest of our paper, we will consider two types of interference: *Internal interference*, which is interference between transmissions from nodes in the network and *external interference*, which is interference from devices outside the network. We argue that a multi-channel protocol can reduce the effects of both types of interference and we will motivate some of our design choices accordingly.

1.1 Internal Interference

Collisions between transmissions are an important cause of loss of throughput in wireless sensor networks. We will argue that even in small or sparse wireless sensor networks, spreading transmissions over multiple channels can greatly reduce this risk. Consider for example Fig. 1. We observe that during a convergecast operation, even though there is a clear bottleneck around the sink node, multiple channels can help reduce (or even completely remove) collisions between transmissions from nodes 2, 3, 4 and 5. Even if the network does not need explicit support for convergecast, it may at some point need to disseminate information (for example, configuration messages) to some or all nodes in the network. During this phase, multiple channel access can reduce the risk of collisions when intermediate nodes forward messages.

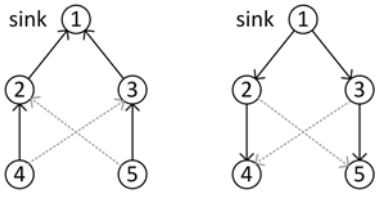


Fig. 1. Congevergecast and dissemination in a simple wireless sensor network

If the total traffic load on a network is low, traffic may be generated in short bursts in time, for example when the network is designed to detect certain rare events. During such a traffic burst, the required bandwidth will be much higher than the average required bandwidth for that network. Moreover, in such cases unsynchronized MAC protocols such as X-MAC and B-MAC suffer from problems with collisions, while synchronized protocols may impose a high overhead since the total traffic volume is very low. Ideally, a multi-channel protocol should be able to increase the available bandwidth while still keeping a low overhead.

1.2 External Interference

Aside from collisions, another common problem is external interference, i.e. interference from sources outside the network. Since WSNs commonly operate

on any of the unlicensed bands, a lot of WSN deployments have to take into account the possibility of other devices operating at the same frequency, such as Bluetooth devices, 802.11 WLAN, baby monitors, cordless phones, etc. Even some models of microwave ovens are known to emit noise on parts of the 2.4GHz band.

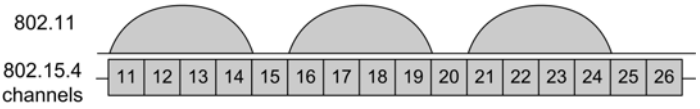


Fig. 2. 802.11 and 802.15.4 channel overlap

Consider for example Fig. 2 where a comparison of 802.11 and 802.15.4 channel spacing is depicted. We could conclude that a simple way to avoid 802.11 collisions is to use one of the four channels which don't overlap between both channel schemes. And indeed, this technique is quite common – for example, in the Contiki OS [5] the default MAC channel is 26 – but this of course only avoids collisions from 802.11 devices. Restricting the number of channels may not be an efficient way to reduce interference from other sources and in fact increases the chance that collocated WSNs will interfere with each other. Ideally, a multi-channel protocol should be able to dynamically switch the channel on which nodes communicate in order to use as much bandwidth as possible, even in the presence of external interference.

2 Existing Multi-Channel Solutions

We will review some of the existing multi-channel MAC techniques to motivate the design choices of our multi-channel MAC protocol. A full overview of existing multi-channel MAC protocols is out of the scope of this paper and can be found in other published work [6,7,8].

First of all, we will limit ourselves to nodes using a single half-duplex radio since this seems to be, to the best of our knowledge, the most common choice in popular sensor node platforms today (e.g. Tmote Sky, JCreate, Mica mote, etc.). We are specifically interested in the way existing techniques (not necessarily directed at sensor networks) assign channels for the nodes in the network.

2.1 Fixed Channel Selection

In fixed channel assignment schemes, nodes are assigned to a fixed channel throughout the lifetime of the network or at least for extended periods during deployment. We can further divide this principle in two categories:

– *Receiver-fixed channel assignment.* Nodes pick a channel to listen to based on a simple algorithm requiring little to no coordination (e.g. channel = ID modulo #channels). This is a simple way to achieve channel diversity

without any coordination overhead. An example of such a protocol is the xRDT protocol described in [7].
- *Coordinated channel selection.* Nodes coordinate channel assignment in such a way that the available bandwidth is distributed efficiently amongst nodes in the network. Protocols like HyMAC [9] and MC-LMAC [8] combine FDMA and TDMA techniques to assign a unique timeslot-channel combination to each node in a two-hop neighborhood. Another example is TMCP [10] which divides a dense WSN in several aggregation trees and assigns a different channel to each tree.

Both type of schemes increase the bandwidth available for communications by increasing the number of channels for unicast transmissions. Simple receiver-fixed assignment offers less advantage in terms of bandwidth, but imposes no coordination overhead on the network. Such a simple scheme could be an elegant solution to increase the bandwidth of uncoordinated MAC protocols such as X-MAC with little implementation overhead.

In general, fixed schemes consider mostly unicast. This restriction can be an issue since broadcast is a commonly required operation used by many WSN routing protocols and applications. Fixed schemes only reduce internal interference and do not consider external interference, so we believe they do not fully exploit the possibilities of multi-channel operation.

2.2 Dynamic Channel Selection

Several techniques have been proposed to allow communication while nodes switch their channels more frequently.

- *Common Hopping.* This is a class of protocols where nodes all listen to the same channel, but "hop" between available channels frequently. Data can be exchanged on a different channel after a handshake on the common channel. Unlike other channel selection schemes, broadcast is easy to support. This type of scheme offers some robustness to external interference, but since packets are quite short in a WSN, the possible bandwidth gain compared to single-channel schemes is limited. To the best of our knowledge, common hopping has not been explored in WSNs for this reason.
- *Independent Hopping.* With independent hopping each node will frequently change its channel, according to its own individual schedule. When a node has data for another node, it switches its radio to the target node's channel and initiates the data transfer (with or without prior handshaking, depending on the protocol). An example of such a protocol is McMAC [11]. This type of scheme has the advantage over common hopping in terms of interference since it protects against both internal and external interference. However, it may be difficult to support broadcast (for example, the McMAC protocol does not explicitly support broadcast transmissions). Additionally, a node must store the hopping schemes of its neighbors. These type of schemes require tight synchronization so nodes can accurately compute the channel of their neighbors.

– *Adaptive channel selection.* Instead of choosing a channel scheme, some approaches also try to adaptively select channels for nodes, based on network density, noise measurements, etc. With this category of schemes, it is theoretically possible to optimize the use of network bandwidth, even in the presence of external interference. In the case of WSNs however, it is not a trivial task to coordinate this accurately over many lossy links at low bandwidth and energy cost. If the coordination fails or crucial packets get lost, nodes could become disconnected from the network.

  An example of a WSN MAC protocol with adaptive channel selection is CoReDac [12] [13] which is designed specifically for convergecast operations. In CoReDac, each node decides on which channel it will receive messages during the next convergecast cycles and reports this in message ACKs to its children. If a node does not receive messages on a certain channel, it will blacklist it and avoid selecting that channel for the next cycles. Although this scheme works well and has a fairly low overhead, it is highly optimized for convergecast and it seems difficult to support other operations.

Another example of dynamic channel selection which combines these techniques is the Y-MAC [14] protocol. In this protocol, time is divided in large frames further divided in slots. The first slots of a frame are reserved for broadcast and control traffic. Each node will then pick one of the remaining slots to listen for possible incoming transmissions. A channel hopping scheme is added to the unicast slots to improve throughput under high traffic loads. This protocol shows good results for high traffic loads (1 message/node/sec and higher in a multi-hop environment), but the coordination between nodes is quite complex and tight synchronization is required. In a real deployment, the authors show this protocol has a duty cycle close to 10% regardless of traffic conditions.

All of the dynamic channel selection schemes presented above have at least some advantage over fixed channel selection with respect to external interference. Even if channel quality is not actively monitored, a dynamic scheme will still provide connectivity since nodes attempt to use several channels. However, a node must be able to accurately compute the channel of its neighbors and this requires either tight synchronization or coordination (or both).

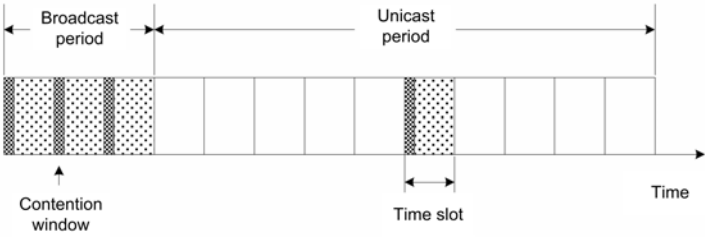


Fig. 3. Y-MAC frame structure, as presented in [14]

2.3 Summary

To conclude this overview, let us summarize some of the general shortcomings of current multi-channel protocols we wish to find an answer to:

- *Fixed channel assignment.* As mentioned above, in a fixed channel assignment scheme, the network does not fully exploit possibilities to increase resilience against external interference. Under external interference parts of the network may be disconnected for certain periods in time.
- *Lack of broadcast support.* Many multi-channel protocols only tackle unicast operations. Even though it may seem contradictory to move all nodes on the same frequency in a multi-channel protocol, broadcasting is a common operation for many routing protocols and applications and should be explicitly supported.
- *Tight synchronization requirements.* Most of the protocols with dynamic channel assignment require tight synchronization of nodes to make sure nodes share the same wake/sleep duty cycle. This way nodes can easily coordinate during their wake-up period. However, synchronization is not a trivial task and can impose a high overhead on the nodes. Typical synchronization protocols require nodes to exchange timestamps. Even if these timestamps can be piggybacked on normal messages, this may not suffice if the network is light on traffic. In such cases, a lot of energy will be spent keeping the nodes synchronized. Out-of-band synchronization such as those described by [15] may also be an option in some cases, but this requires additional hardware and thus an additional cost which is not negligible.

In general, we can also conclude that there appears to be a strong correlation between protocol features and protocol requirements. As a protocol adds more features, the requirements in terms of coordination and synchronization quickly increase as well. We wish to create a protocol which exploits the possibilities of multi-channel operation without imposing strong requirements on the network.

3 MAC Design

With these considerations described above in mind, these are the requirements for our Multi-Channel MAC (MuChMAC) protocol:

- *Frequency agility*: The nodes should use multiple channels dynamically to increase bandwidth and to allow the network to continue operation even if there is external interference.
- *Broadcast support*: The nodes should be able to efficiently send broadcast messages.
- *Low power*: The protocol should be optimized in such a way that nodes can operate under low duty cycles (only a few %) to increase battery lifetime.
- *General purpose*: The protocol should be able to function sufficiently well (in terms of energy consumption, latency, bandwidth, ...) under a variety of traffic loads and patterns. We will target traffic loads of one message per node every few seconds, down to a few messages per node per hour.

3.1 Frequency Hopping with Broadcast

Let us tackle the first issues: How do we design our protocol to be frequency agile, yet give it broadcast capabilities? To achieve this, we will divide time in slots and let each node switch its radio frequency every slot. We base our assignment of slots on the *parallel rendez-vous* principle, as proposed for McMAC [11]: To calculate the radio channel of a node, we input its ID and the current slot number in a pseudo-random generator. The channel number is chosen by the receiver; when a node has a message for another node, it switches its radio to the receiver channel and sends the message. By choosing a pseudo-random hopping scheme, we avoid that nodes have to store the hopping scheme of all their neighbors.

We will discuss further on how a node will be able to calculate the slot number without synchronization overhead. First, let us propose the adaptation presented in Fig. 4 to add broadcast support to this channel hopping scheme.

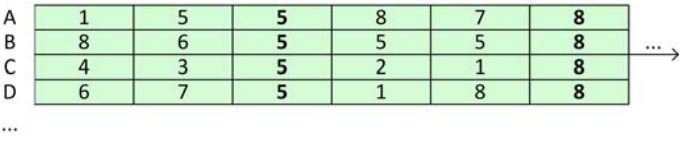


Fig. 4. Parallel rendez-vous scheme extended with broadcast slots

This scheme is partially inspired by the Y-MAC [14] frame structure. Every u/b unicast slots, a broadcast slot will be inserted. These broadcast slots also follow a pseudo-random hopping sequence, but they are the same for each node. This way, our hopping scheme is a combination of independent and common hopping schemes, trying to take the best of both worlds. Figure 4 demonstrates this principle with $u/b = 2$ for the channel selection of 4 nodes A, B, C and D over time. Other values of u/b can be chosen during network setup if the relation of unicast and broadcast traffic is known in advance.

One disadvantage of this approach is that the hopping scheme does not actively adapt to interference on certain channels but instead gives us a more passive interference avoidance; if there is interference on some channels we still have connectivity on the rest of the channels. This is not as powerful as active avoidance, but it does not require any coordination overhead. Accordingly, we motivate our choice with the observation that coordinating channel selection among nodes in the network is not a trivial task and may cause a high overhead when actual traffic volumes are low, thus violating our "General Purpose" constraint.

3.2 Low-Power Operation

The original parallel rendez-vous scheme was designed targeting high-power nodes, such as 802.11 WLAN routers or laptops. It is obvious that this scheme,

where radios are assumed to be listening to the medium at all times, is not applicable to low-power sensor nodes. In order to achieve the low power constraint, we will put each node in a low sleep/wake duty cycle. As shown in Fig. 5 nodes will wake up only during a small portion of a slot and will stay asleep for the rest of the slot. For example, in Fig. 5 a duty cycle of 20% is shown. In our implementation we set a duty cycle of 1%.

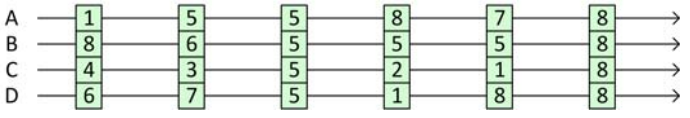


Fig. 5. Low duty cycle

3.3 Synchronization

As discussed earlier, keeping nodes in the state depicted in Fig. 5 is not a trivial task and will require tight synchronization, especially if very low duty cycles are desired. However, we could ask ourselves if this synchronization is really a necessity. For example, starting from the situation depicted in Fig. 5, let us consider how this situation evolves if we don't synchronize. Assuming a slot size of 500ms and a drift of at most ±40 ppm (as specified in the 802.15.4 PHY standard), how the situation might look like after 30 minutes is depicted in Fig. 6. It is clear in the worst case the wake-up periods have drifted away from each other. But if we look at the slot transitions (as defined by an absolute master clock with zero drift) depicted as dashed lines in Fig. 6, we observe that the wake-up periods have not drifted past these boundaries yet, despite long unsynchronized operation. So if we know when another node wakes up, we can still correctly calculate the slot number and the associated channel number. Unfortunately, it is impossible to calculate the exact time when a node will wake up. If we try to calculate this, our estimate will have a resolution equal to the upper limit on drift time as given by (1).

$$\Delta t_{\max} = \left(t_{(\text{now})} - t_{(\text{last synch})}\right) \times \text{drift}_{\max} \tag{1}$$

Typically, this will be many times the wake-up period of a node, so it is clear we can only calculate a period during which a target node will wake up, but not

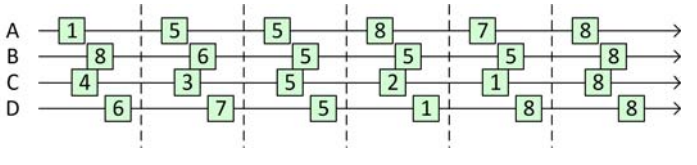


Fig. 6. Unsynchronized operation

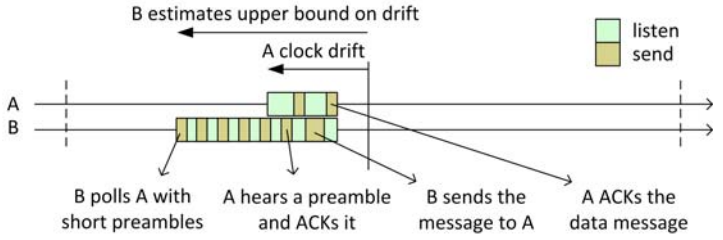


Fig. 7. Unicast communication between nodes

the exact moment. To overcome this problem, we will have nodes communicate according to an adaptation of the X-MAC scheme as depicted in Fig. 7: When a node B has a message for another node A, B will first calculate the lower limit of the next period when that node will be awake. At that point, node B will calculate the channel of node A and will start sending out small preamble messages. When A wakes up, it will hear a preamble, and it will acknowledge to B that it is awake and that the data can be sent. After the data message has been sent, A can optionally acknowledge the data message. Broadcasting will happen according to a roughly similar scheme, except that the sending node will repeatedly send the data message instead of preambles and there is no feedback from receiving nodes, as depicted in Fig. 8.

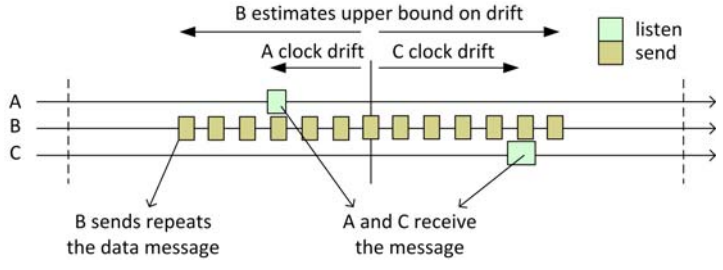


Fig. 8. Broadcast communication

Now the final problem we have to tackle is to make sure nodes do not desynchronize beyond slot transitions. This however only requires very loose synchronization. With the above scheme we must make sure drift is always smaller than half the slot size. With some typical parameters we get:

$$\text{synch period} = \frac{\text{slot}/2}{\text{drift}} = \frac{250\,\text{ms}}{80\,\text{ppm}} = 3125\,\text{s} = 52\,\text{min} \tag{2}$$

Not only does this mean we have a very large synchronization period, it also means that when we synchronize, we do not need very strict synchronization (i.e. down to a few clock cycles) but down to an order of magnitude of $1 - 10\,\text{ms}$ is

more than sufficient. Moreover, the entire network does not need to be globally synchronized, it is sufficient that a node is synchronized with its neighbors. This allows us to use simple timestamp based synchronization, where nodes put timestamps on all outgoing and incoming messages and calculate clock offsets accordingly. In the case of unicast transmissions, it is even sufficient to timestamp the preamble and ack messages, causing no overhead on the data message itself. With such an approach, the network will also be partially adaptive to traffic: when messages are sent more frequently, the drift against neighboring clocks will be smaller and less energy will be wasted sending preambles.

Intermezzo: The Contiki `timesynch` Module. Contiki has a module called `timesynch` which is implemented for all platforms using a CC2420 radio. This module adds code to the radio driver that timestamps all outgoing and incoming packets. The network should also have at least one master node with an "absolute" clock. Such a node will be defined to have an *authority* of 0. The neighbors of a master will have authority 1, their neighbors 2 and so on (depicted in Fig. 9). To create this "authority gradient", the following mechanism is implemented: When a non-master node receives a packet from another node with lower authority a, it will set its own authority to $a+1$ and will synchronize to that node, using the timestamps from the received packet. This way, nodes will synchronize with neighbors closer to a master clock.

In our implementation we have used this algorithm with a small addition to estimate an upper bound on clock drift against neighboring nodes.

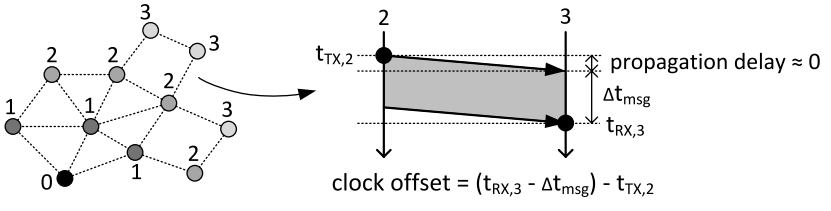


Fig. 9. A `timesynch` network with one master, illustrating clock offset calculation between two nodes with different authority

3.4 TDMA Optimization

When density increases, the chances increase that multiple nodes select the same frequency during the same unicast slot. For example, assuming a channel is picked randomly out of 8 orthogonal channels, two nodes will have a 1/8 chance of picking the same channel. A node with 5 two-hop neighbors only has a 48% chance of picking a unique channel and if a node has 10 two-hop neighbors, there will always be at least 3 nodes on the same channel with each node having a 26% chance of a unique channel. If nodes would wake up at the same time in a slot, this would reduce the amount of parallel traffic that can be achieved

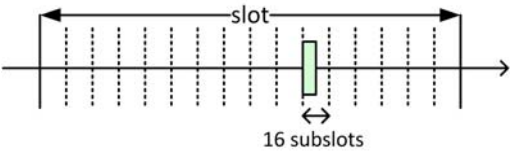


Fig. 10. TDMA timing optimalization

on the network, which was exactly one of the benefits we were looking after. To overcome this problem, we will design the timing of our slot as depicted in Fig. 10. Instead of waking up in the middle of a slot, we will split slots in a number of "subslots" which consist of a small wake-up window for a node and a small guard period to the next subslot. The subslot in which a node will turn on its radio is determined by its ID and the current slot number. For example, in our implementation, we have chosen 16 subslots, giving each subslot a wake-up of 5 ms and a guard of 26 ms. Looking back at the previously mentioned scenarios, two nodes now only have a 1/128 chance of picking the same channel-timeslot combination. Nodes with 5 and 10 two-hop neighbors have a 96% and 92% chance respectively of picking a unique channel-subslot combination for any given slot. With this design, MuChMAC will behave like a multi-channel TDMA protocol under high traffic loads, while behaving more like a multi-channel X-MAC under low traffic loads.

In some TDMA protocols such as MC-LMAC [8] or Y-MAC [14], it is specified that a node can take multiple timeslots in one frame. This way nodes can trade off power against bandwidth and latency. We will add this feature to our TDMA optimization: depending on the "power level" of a node, that node will pick one or more subslots in each slot. The slots are picked in such a way that all slots for a power level p are also picked for $p+1$, as depicted in Fig. 11. A node will still be able to communicate with a neighbor even if it's value for the power level of that neighbor is too low. If a node does not know the power level of a neighbor, it can simply pick $p = 1$. In our implementation, we add the power level of a node to packet headers and we store all received power levels in a neighbor table. After packet loss, we decrement the stored value of power level for a node, with a minimum of 1. This way, the overhead of communicating power level is small (16 power levels = 4 bits in packet header) and no additional coordination packets have to be sent.

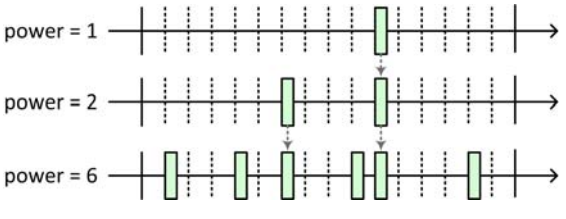


Fig. 11. Subslot selection for different power levels

Table 1. MAC evaluation parameters

MuChMAC	
slot	500 ms
wake-up	5 ms
subslots	16
channels	8
unicast : broadcast	2 : 1
X-MAC	
slot	500 ms
wake-up	5 ms

4 Experiments

We will perform our experiments on a simple testbed consisting out of Sentilla JCreate motes. The network topologies are shown in Fig. 12. We consider a 5-node and 7-node setup, with nodes arranged in two parallel lines to the sink. The nodes are arranged in such a way that transmissions from any node can interfere with transmissions from another node. We use a simple routing layer which forwards all received or generated messages to a predefined parent node. When a message could not be sent – if the receiving node was sending, or if there was contention with another node – the message is retransmitted with a random back-off, with a maximum of 4 retransmissions per message. To improve throughput under higher loads, we added a packet queue of 4 messages to the routing layer. A packet is considered lost when all retransmissions have failed or when it is dropped because of a full packet queue.

We will compare our MAC layer with X-MAC, the standard asynchronous Contiki MAC layer. The parameters for both layers are set up as shown in Table 1. We have chosen to use only 8 out of the 16 available channels since several authors have demonstrated that channels in the 802.15.4 band may experience interference from adjacent channels [8,10]. X-MAC uses its default

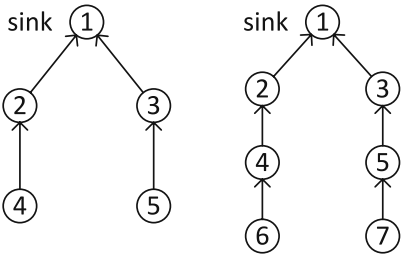


Fig. 12. Testbed setups

channel 26. The network is set up in an office environment. We detected some occasional interference on 802.11 bands, but we did not purposely cause any external interference during our experiments.

First, we will test the throughput of the network when each node generates messages for the sink at a constant rate. The inter-message period is slightly randomized to reduce any scheduling effects. Secondly, we will have a look at the network throughput when messages are generated in event bursts, i.e. several nodes in the network generate messages at about the same time, but with long inter-message periods. For each experiment we will also have a look at the activity of a node, defined as the % of time the radio is on (for transmitting, receiving or idle listening).

$$\text{activity} = \frac{t_{\text{tx}} + t_{\text{rx}} + t_{\text{listen}}}{t} \tag{3}$$

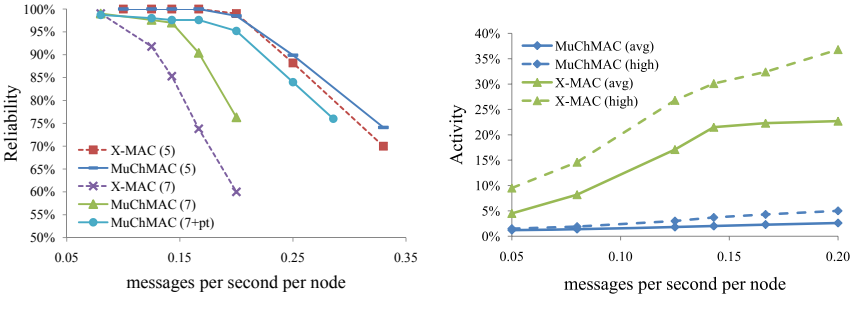
This should give us a good measure for the energy use of the nodes.

4.1 Constant Network Load

To test throughput, we will measure the reliability of the the network, as defined by (4), under increasing network loads.

$$\text{reliability} = \frac{\sum \text{messages generated}}{\text{messages received by sink}} \tag{4}$$

The results of these experiments are represented in Fig. 13. From our experiments we observed that quite often a packet would wait in a queue while another is waiting at the MAC layer to be sent to the same receiver. From this observation, we implemented a "packet train" optimization: when several packets have to be sent to the same receiver, MuChMAC will send the second packet immediately



(a) Reliability vs Message Rate (5 and 7-node networks)

(b) Activity vs Message Rate (7-node network)

Fig. 13. Experimental results

after the first has been ACK'ed, repeating until no more packets can be sent to that receiver. The results from this optimization are marked '+pt' on Fig. 13(a). For the 5-node network, we can observe MuChMAC offers a only a small increase in reliability. The gain here is limited because there are not so many parallel transmissions and some of the multi-channel bandwidth gain is lost because only 2 out of 3 slots are reserved for unicast.

In the case of the 7-node network, the difference is more significant. Under moderate traffic loads, X-MAC is losing packets on all links, while MuChMAC only loses packets on the links to the sink. As expected, the packet train optimization significantly improves reliability under higher traffic loads, keeping $> 90\%$ reliability for 0.2 packets per second per node ($= 1.2$ packets per second overall).

Figure 13(b) shows average activity and activity of the most active node for the interesting range of traffic loads in the 7-node network. The results for activity were similar in both versions of MuChMAC with or without packet trains. We observe that activity is much lower in the case of MuChMAC, around $1-5\%$, whereas X-MAC activity reaches $10-30\%$ under moderate traffic loads. This is of course because MuChMAC is synchronized when packets are sent or received, so the number of preamble packets can be greatly reduced when there is a lot of traffic. For very low constant traffic loads X-MAC and MuChMAC show similar behaviour without significant differences.

These results demonstrate the applicability of MuChMAC over a wide range of traffic loads and show that it will maintain low power operation even under moderate stress. We can conclude that MuChMAC fulfills the requirements we described in paragraph 3.

4.2 Traffic Bursts

In the previous paragraph, we have shown that MuChMAC offers comparable performance to X-MAC when traffic loads are small. This was tested however with nodes randomly producing packets. In the following experiments, we tested the performance of both MAC layers when all nodes (except the sink) in the network generate traffic in bursts: all nodes generate one message, spread over at most 0.5 s, with an inter-message period of two minutes (total load ≈ 0.008 message per second per node).

The results of these experiments are shown in Table 2. We also present average ($\pm$ standard deviation) latency from source to sink for all packets. Since we have given each node fairly large packet queues, nodes have sufficient buffer space to hold incoming messages while transmitting. With this setup reliability is high and comparable for both MAC layers and each protocol is able to consistently deliver at least 5 out of 6 messages to the sink for each traffic burst. However, X-MAC will need more transmissions per message on average and this has an impact on activity and latency. X-MAC activity is still low since bursts are spread out in time, but it is significantly higher than MuChMAC. Average latency to reach the sink is also considerably higher in the case of X-MAC. For MuChMAC, the packet train optimization offers a significant advantage without an additional

Table 2. Traffic burst experimental results for the 7-node network

	MuChMAC	MuChMAC+pt	X-MAC
msg/node	0.008/s	0.008/s	0.008/s
reliability	99%	99%	99%
activity	1.2%	1.2%	2.5%
latency	3.3 ± 2.4s	2.3 ± 1.8s	4.3 ± 2.7s

energy cost. These results demonstrate that MuChMAC is better fit to handle the traffic bursts generated by the network.

5 Conclusion and Future Work

These experiments on a real testbed demonstrate the applicability and usefulness of MuChMAC and show that it can efficiently exploit multi-frequency operation without coordination or synchronization overhead. We have shown that the multi-channel operation provided by MuChMAC can be useful even for small networks with low overall traffic load. The simple scenarios we evaluated give a reasonable first-order impression of its performance. In future work we will use testbeds containing more nodes. Furthermore, in paragraph 3.4, we have proposed a mechanism for nodes to adjust their performance using a "power level" parameter. We will study how a node can choose its power level depending on its energy supply, required bandwidth, etc. in an intelligent way. Finally, we will carry out experiments to test the performance of MuChMAC when controlled external interference is present.

Acknowledgements. This work was done in the scope of FWO project G.0291.09N and in preparation for tests with the Hercules project equipment.

References

1. Van Dam, T., Langendoen, K.: An adaptive energy-efficient mac protocol for wireless sensor networks. In: SenSys 2003: Proceedings of the 1st international conference on Embedded networked sensor systems, pp. 171–180. ACM, New York (2003)
2. van Hoesel, L., Havinga, P.: A lightweight medium access protocol (LMAC) for wireless sensor networks. In: 1st International Workshop on Networked Sensing Systems (INSS), Tokio, Japan, Society of Instrument and Control Engineers (SICE), pp. 205–208 (2004)
3. Polastre, J., Hill, J., Culler, D.: Versatile low power media access for wireless sensor networks. In: SenSys 2004: Proceedings of the 2nd international conference on Embedded networked sensor systems, pp. 95–107. ACM, New York (2004)

4. Buettner, M., Yee, G., Anderson, E., Han, R.: X-MAC: A short preamble mac protocol for duty-cycled wireless sensor networks. Technical report, Department of Computer Science University of Colorado at Boulder (2006)
5. 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), Tampa, Florida, USA (2004)
6. Mo, J., So, H.-S.W., Walrand, J.: Comparison of multi-channel mac protocols. In: MSWiM 2005: Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems, pp. 209–218. ACM Press, New York (2005)
7. Maheshwari, R., Gupta, H., Das, S.R.: Multichannel mac protocols for wireless networks. In: SECON 2006: 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks, vol. 2, pp. 393–401 (2006)
8. Incel, O.D.: Multi-channel wireless sensor networks: protocols, design and evaluation. PhD thesis, University of Twente, Enschede (2009)
9. Salajegheh, M., Soroush, H., Kalis, A.: Hymac: Hybrid tdma/fdma medium access control protocol for wireless sensor networks. In: IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp. 1–5 (2007)
10. Wu, Y., Stankovic, J., He, T., Lin, S.: Realistic and efficient multi-channel communications in wireless sensor networks. In: INFOCOM 2008. The 27th Conference on Computer Communications, pp. 1193–1201. IEEE, Los Alamitos (2008)
11. sheung Wilson So, H., Walr, J.: McMAC: A multi-channel mac proposal for ad-hoc wireless networks. Technical report, In Proc. of IEEE WCNC 2007, Hongkong (2005)
12. Österlind, F., Voigt, T.: CoReDac: Collision-free command-response data collection. In: Proceedings of 13th IEEE International Conference on Emerging Technologies and Factory Automation, Hamburg, Germany (2008)
13. Voigt, T., Österlind, F., Dunkels, A.: Improving sensor network robustness with multi-channel convergecast. In: Proceedings of 2nd ERCIM Workshop on e-Mobility, Tampere, Finland (2008)
14. Kim, Y., Shin, H., Cha, H.: Y-MAC: An energy-efficient multi-channel mac protocol for dense wireless sensor networks. In: IPSN 2008: Proceedings of the 7th international conference on Information processing in sensor networks, Washington, DC, USA, pp. 53–63. IEEE Computer Society, Los Alamitos (2008)
15. Rowe, A., Mangharam, R., Rajkumar, R.: RT-Link: A global time-synchronized link protocol for sensor networks. Ad Hoc Networks 6, 1201–1220 (2008); Energy Efficient Design in Wireless Ad Hoc and Sensor Networks

Exploiting Overlapping Channels for Minimum Power Configuration in Real-Time Sensor Networks

Xiaodong Wang[1], Xiaorui Wang[1], Guoliang Xing[2], and Yanjun Yao[1]

[1] Department of EECS, University of Tennessee, Knoxville, TN 37996
{xwang33,xwang,yyao9}@utk.edu
[2] Department of CSE, Michigan State University, MI 48824
glxing@cse.msu.edu

Abstract. Multi-channel communications can effectively reduce channel competition and interferences in a wireless sensor network, and thus achieve increased throughput and improved end-to-end delay guarantees with reduced power consumption. However, existing work relies only on a small number of orthogonal channels, resulting in degraded performance when a large number of data flows need to be transmitted on different channels. In this paper, we conduct empirical studies to investigate the interferences among overlapping channels. Our results show that overlapping channels can also be utilized for improved real-time performance if the node transmission power is carefully configured. In order to minimize the overall power consumption of a network with multiple data flows under end-to-end delay constraints, we formulate a constrained optimization problem to configure the transmission power level for every node and assign overlapping channels to different data flows. Since the optimization problem has an exponential computational complexity, we then present a heuristic algorithm designed based on Simulated Annealing to find a suboptimal solution. Our empirical results on a 25-mote testbed demonstrate that our algorithm achieves better real-time performance and less power consumption than two baselines including a scheme using only orthogonal channels.

1 Introduction

Many wireless sensor network (WSN) applications must address multiple stringent design constraints such as energy consumption and end-to-end communication delay. Energy has long been treated as the primary optimization goal for battery-powered wireless sensor nodes. With lower energy consumption, a network can achieve a longer lifetime. In addition to periodic sleeping, one of the effective ways to reduce node energy consumption is to lower its radio transmission power. This can be supported by the existing sensor mote hardware. For example, the CC2420 radio chip [1] used in many mote hardware platforms has 31 different transmission power levels. However, reducing transmission power may lead to unreliable wireless links and cause increased number of retransmissions. As a result, it may lead to poor guarantees of other important design

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 97–113, 2010.

constraints such as end-to-end delay, as many WSN applications require information to be transmitted from sources to sinks within an application-specified deadline. Therefore, transmission power must be carefully configured in order to meet the desired constraints of a WSN. High transmission power may improve the quality of a single wireless link but may lead to increased power consumption, stronger interferences to other links, and reduced network capacity [2].

The emergence of multi-channel mote hardware has made it possible to achieve improved throughput and delay guarantees with reduced transmission power, by using different channels on different nodes, leading to less channel competition and interference in the network. For example,the CC2420 radio chip provides 16 wireless channels with radio frequency from 2,400 to 2,483MHz. As a result, multi-channel communication protocols have been proposed for WSNs to improve the performance of traditional single-channel protocols commonly used in WSNs. Based on the channel allocation scheme, existing multi-channel protocols can be categorized to two classes: node-based and flow-based. In node-based protocols, channels are assigned to different nodes locally to minimize interferences. For example, several node-based multi-channel MAC protocols [3] [4] have been proposed to improve network throughput for WSNs. However, a major problem for node-based assignment is that nodes usually need to switch channels in order to receive data from and transmit to different neighbors, which may result in a high overhead, in terms of latency and power consumption. In flow-based protocols, the nodes in the same data flow are assigned the same channel so that frequent channel switching is avoided. For example, a flow-based multi-channel real-time communication protocol, known as MCRT [5], has recently been presented to allow different data flows to transmit on different channels for improved end-to-end real-time guarantees with reduced power consumption. MCRT has been demonstrated to outperform node-based schemes by having a smaller deadline miss ratio and lower power consumption.

While multi-channel communications have shown great promise, recent studies (*e.g.*, [6]) conducted experiments on Micaz hardware to investigate multi-channel realities in wireless sensor networks. An important reality reported is that the number of orthogonal channels is actually small for the existing mote hardware. Accordingly, it has been suggested that a practical multi-channel communication protocol should only rely on a small number of non-adjacent orthogonal channels, because adjacent overlapping channels may have undesired inter-channel interferences. For example, at most, only 8 channels out of the 16 channels provided by the CC2420 radio chip can be used as orthogonal channels [6], resulting in the waste of half of the available wireless channel resources. While 8 channels may be enough for some WSN applications, using only orthogonal channels has limited the further improvement of network throughput, real-time performance, and power optimization in the commonly used many-to-one traffic pattern, where the number of data flows can be large in a network.

In this paper, we propose to utilize adjacent overlapping channels to configure power and channels for a WSN to achieve improved real-time performance and reduced power consumption. The power and channel configuration problem is

defined as follows: Given a WSN with multiple data flows from different sources to the base station, our goal is to assign channels (including overlapping channels) to the data flows and determine a transmission power level for every node in the network, such that the overall (transmission) power consumption of the network can be minimized while the average end-to-end delay of each data flow can be guaranteed to stay within a deadline. In order to motivate our work, we first conduct hardware experiments to investigate the interferences among overlapping channels. We then use empirical studies for overlapping channel modeling. Based on our models, we formulate the power and channel configuration problem as a constrained optimization problem, with power minimization as the objective and the end-to-end delay as the constraints. Since it is cost-prohibitive to find the optimal solution, we propose a heuristic algorithm based on Simulated Annealing to find a suboptimal solution. Finally, we conduct experiments on a 25-mote testbed to show that our configuration outperforms two baseline solutions.

To our best knowledge, this paper presents the first study of utilizing overlapping channels to achieve minimum transmission power configuration and guaranteed real-time performance in wireless sensor networks. Specifically, the contributions of this paper are four-fold.

- We conduct empirical studies to investigate the interferences among overlapping channels. Our results show that overlapping channels can also be utilized for improved real-time performance if the node transmission power is carefully configured.
- We establish an empirical model between received signal strength (RSS) and transmission power level for overlapping channels. Based on the RSS model, we model the relationship between packet reception ratio (PRR) and RSS to account for the interferences from overlapping channels.
- We formulate the power and channel configuration problem as a constrained optimization problem. Since the problem has an exponential computational complexity, we then present a heuristic algorithm designed based on Simulated Annealing (SA) to find a suboptimal solution, which can be executed periodically or in an on-demand fashion.
- We implement our algorithm on the Tmote hardware and conduct experiments on a 25-mote testbed. Our results demonstrate that our algorithm can reduce both end-to-end communication delay and overall transmission power consumption, compared with two baselines. The first baseline conducts the same optimization using only orthogonal channels. The second baseline uses SA to find the desired power level but randomly assigns overlapping channels.

The rest of paper is organized as follows. Section 2 highlights the distinction of our work by discussing the related work. Section 3 presents empirical studies to motivate this work and build models for overlapping channels. Section 4 introduces the formulation of our optimization problem with real-time performance analysis. Section 5 presents the algorithm we used to solve the optimization problem. In Section 6, we present the empirical results of our algorithm. Section 7 concludes the paper.

2 Related Work

Recent studies have proposed to use partially overlapping channels (POC) in wireless mesh networks. Liu et al. [7] propose a channel allocation scheme for link scheduling, which takes advantages of POC to obtain better throughput for mesh networks. Feng et al. [8] establish an interference model for POC-based wireless networks, and use numeric methods to improve overall network capacity. A linear model for channel assignment, which uses a channel overlapping matrix and mutual interference matrices to model POC channels, has been proposed in [9]. However, no detailed study has been performed to utilize overlapping channels in wireless sensor networks for improved real-time performance. A coarse-grained channel assignment policy for WSNs is proposed in [6], which allocates non-overlapping channels to disjoint trees and exploits parallel transmissions among trees. In this paper, we propose to utilize overlapping channels to configure power and channels for a WSN to achieve better real-time performance and energy efficiency.

Several projects have studied received signal strength (RSS) and its utilization in WSNs. Sha et al. [10] establish a model between the RSS and transmission power in a single channel. Demirbas et al. [11] present a robust and lightweight solution for sybil attack problem based on the received signal strength indicator (RSSI) readings of messages in WSNs. However, none of these projects study the relationship between RSS and transmission power in multi-channel WSNs. In this paper, we establish the RSS model for overlapping channels and formulate a transmission power minimization problem based on the models.

Many real-time communication protocols have been proposed for wireless sensor and ad hoc networks. A comprehensive review of real-time communication in WSNs is presented in [12]. At the MAC layer, Implicit EDF [13] is a collision-free real-time scheduling scheme by exploiting the periodicity of WSN traffic. At higher layers, SPEED [14] achieves desired end-to-end communication delays by enforcing a uniform communication speed throughout the network. However, most of the existing real-time protocols do not take advantage of the capability of multi-channel communications available in today's mote hardware. In our work, we address the problem of utilizing overlapping channels for improved real-time performance.

Different from all the aforementioned work that handles real-time guarantees, partially overlapping channels, and energy efficiency in isolation, our design utilizes overlapping channels available on existing sensor mote hardware to achieve more energy-efficient transmission for multi-channel WSNs under real-time constraints.

3 Empirical Modeling of Overlapping Channels

Previous work [6] has reported that adjacent overlapping channels have undesired inter-channel interferences. In this section, we first investigate the impacts of overlapping channels on packet reception ratio (PRR) of link to motivate our

work. We then extend existing work to establish an empirical model between RSS and transmission power level for overlapping channels. Based on the RSS model, we derive a PRR model to account for the interferences from overlapping channels.

3.1 Case Study for Motivation

In this section, a case study is performed with two pairs of nodes, which compose two one-hop communication links. In the experiment, one pair of motes performs as the transmission pair by using channel 16. The other pair of motes, acting as the jammer pair, uses an adjacent channel, 15, to communicate. The transmissions of these two pairs are synchronized. The transmission power of the transmission pair is fixed at power level 15, while the transmission power of the jammer pair increases one level at a time from level 3 to level 31. One hundred packets are transmitted on both pairs at each power level. We calculate the PRR of each pair in this experiment under different transmission power levels of the jammer pair. The results are shown in Figure 1. From the results we can see that both the two pairs can achieve a good PRR when the jammer pair is using power levels 16 to 18 for transmission. When the jammer is using a lower power level, its communication does not incur much interference to the transmission pair. The transmission pair can reach a high PRR. When the jammer pair is using a higher power level to transmit, it can improve the packet reception ratio of its own communication, but incurs too much interference to the transmission pair, and so hurts the communication quality of the transmission pair. This experiment shows that given two communication links working on overlapping channels, we can achieve good quality for both transmissions if we carefully choose the transmission power.

3.2 Overlapping Channel RSS Model

As discussed in subsection 3.1, with careful selection of transmission power, two links working on adjacent channels can both achieve a high PRR. An approximate linear correlation between RSS and transmission power over a

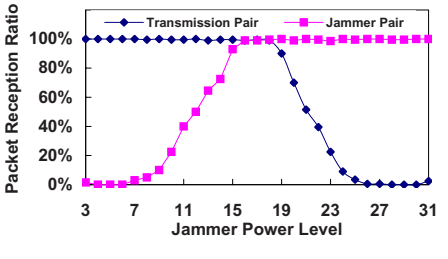


Fig. 1. Packet reception ratio vs. jammer power level. (Jammer uses channel 15 and sender uses channel 16.)

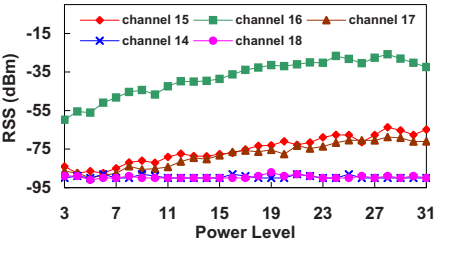


Fig. 2. RSS vs. power level on different channels. (Sender uses channel 16.)

single-channel single-hop link is reported in [10]. In this subsection, we extend the method proposed in [15] to study the relationship between RSS and transmission power in the scenario where a sender and a receiver are working on adjacent channels. We conduct the signal strength detection experiment on a single link to explore the overlapping channel property.

Our experiment uses two Tmote Invent motes. One mote acts as the sender and the other as the RSS sensor. In the experiment, the sender continuously broadcasts packets at a rate of 100 packets per second. The RSS sensor continuously collects the received signal strength by periodically reading the value of the Received Signal Strength Indicator (RSSI) on the mote at a rate of 100 times per second. After sending 100 packets at one power level, the sender lowers its transmission power by 1 level, starting from level 31 to level 3. We first filter out the noise value by using the noise floor threshold we collected before the experiment and then calculate the average RSS value. We test various combinations of sending and receiving channels in this experiment.

Figure 2 shows the result when the sender is using channel 16. We can see that when the sender is using channel 16 for broadcasting, the RSS values sensed on the two adjacent channels, channel 15 and channel 17 show highly linear correlation with sender's transmission power. However, no clear RSS reading is sensed on channel 14 or channel 18. In addition, the results show an approximate linear increasing trend when the sender and the RSS sensor are using the same channel, channel 16. Previous work [10] presents the empirical single-channel RSS-Power model as:

$$RSS(v, u, p_u) = A_{u,v} \times p_u + B_{u,v}, \tag{1}$$

where v is the receiving node, u is the sending node, and p_u is the transmission power at u. A and B are two parameters of the model, which can be calculated by applying linear curve fitting to the sampled data. Note that distance is not considered in Equation 1 because the RSS value is dynamically measured between each given pair of sender and sensor.

Based on the observation of similar linear pattern when the sender and receiver are using adjacent channels, we re-establish the empirical RSS-Power model under multi-channel conditions as:

$$RSS(v, u, p_u, c_v, c_u) = A_{u,v,c_v,c_u} \times p_u + B_{u,v,c_v,c_u}, \tag{2}$$

where c_u is the transmitting channel for sender u and c_v is the listening channel for receiver v. A and B are the two model parameters, which are usually decided by the the application environment, such as network condition and communication distance. A similar model is reported in [15]. Our model uses a simplified threshold filter to filter out the noise for faster runtime processing, while a CPM noise filter is used in [15].

Using linear curve fitting to establish our model gives us a fast way to accomplish the model establishment, depending on the number of sampling points we need. One second is required for the signal strength readings for each power level in the model, as explained previously. If we use 5 power levels to build the

model, the total time for the model establishment is only 5 seconds. Therefore, our model can be promptly rebuilt at runtime to adapt to environmental or temporal variations of network conditions. Also, the overlapping channel RSS model of every node in a less dense network (*e.g.*[16]) can be quickly established.

3.3 Packet Reception Ratio

Packet reception ratio (PRR) is the probability that a packet can be received successfully. Higher transmission power can provide a higher Signal to Interference and Noise Ratio (SINR) over the link, which leads to a higher PRR. However, with higher transmission power, the communication at the current link could significantly interfere with another link's communication as shown in subsection 3.1. In this section, we conduct an experiment to study the relationship between PRR and SINR. With an understanding of this relationship, we can find the appropriate transmission power range to reach a required SINR value for a desired PRR value.

In the experiment, we use three Tmote Invent motes, one as the receiver C and the other two as transmitting motes, A and B. All of the three motes use the same channel. This experiment consists of three rounds. In the first round, we only turn on motes A and C. We use A to transmit multiple packets to receiver C and calculate the average received signal strength of the packets, denoted as $RSS(A, C)$. In the second round, we turn A off and use B to transmit multiple packets to the receiver. We then calculate the average received signal strength of B, denoted as $RSS(B, C)$. In the third round, all the three motes are turned on. Both A and B transmit multiple packets to receiver C. The transmissions are synchronized. We calculate the PRR for A's transmission, denoted as $PRR(A, C)$.

Considering B's transmission as the interference to A's transmission, we can calculate the SINR value for A's transmission as follows:

$$SINR(A, C)_{dB} = RSS(A, C) - 10log_{10}(10^{\frac{RSS(B,C)}{10}} + 10^{\frac{N}{10}}) \quad (3)$$

where N is the noise floor value, which is collected before the experiment. Equation 3 is derived from the SINR equation from [17]. By doing the above three steps and applying the equation, we get a PRR-SINR pair. We repeat the experiment with different distances from B to receiver C and different transmission power levels used by A and B to create different SINR values at the receiver. Figure 3 shows the PRR-SINR relationship in our experiment. When the SINR value is greater than 6dB, the PRR is almost 100%. Therefore, in order to achieve a good packet reception ratio in this sepcific experiment, *e.g.*,

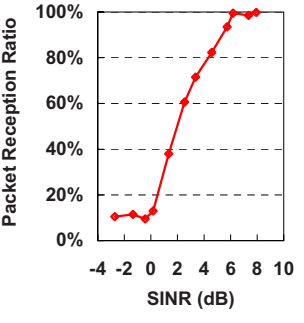


Fig. 3. PRR vs. SINR

90%, we need to choose a transmission power that can provide a strong enough received signal strength leading to an SINR value of more than 6dB.

In order to apply this experimental approach to multi-channel networks, we extend the PRR-SINR relationship by incorporating the channel information to it. We use ($SINR_v$, c_v, PRR_v), to denote the PRR-SINR-Channel relationship between node v's packet reception ratio and the corresponding SINR value in channel c_v. With this extension, we can obtain the required SINR value for a good PRR on a desired channel.

4 Minimum Transmission Power Configuration

In this section, we first formulate the power and channel configuration problem. We then analyze the node transmission delay in the network.

4.1 Problem Formulation

We assume the network has the common many-to-one traffic pattern [18] [19], which is composed of multiple sources, some relay nodes and one base station. Each source generates a data flow to the base station. All the flows are assumed to be disjoint, since disjoint paths are widely used in multi-path routing to enhance the system's fault-tolerance [5] [20]. The data generated at the source are assumed to follow a uniform random distribution [21]. We also assume that the base station is a super node with multiple radios such that it can work on several different frequencies at the same time. The channel allocation in our network is flow-based, which means all nodes in the same flow work on the same channel. Our goal is to minimize the total transmission power consumption under the constraint that the end-to-end delay of every flow in the given topology is constrained.

We first introduce the following notation:

- $G = (V, E)$, a directional graph denoting the network with V nodes and E edges (links).
- f_i, the data flow with the id number i.
- D, the delay constraint for each flow.
- p_u, transmitting power used by node u.
- c_u, the channel id used by node u, which is an integer number.
- $I(v)$, the interference node set of node v.
- (u, v), a communication link in the graph, in which u is the sending node and v is the receiving node.

Given the notation above, we can formulate our minimization problem as:

$$min \sum_{v \in V:(u,v) \in G} p_u \times \frac{1}{PRR(u,v)} \tag{4}$$

Subject to the constraints:

$$1 \leq c_u \leq n \qquad \forall u \in G \tag{5}$$

$$c_u = c_v \qquad \forall (u,v) \in G \tag{6}$$

$$\sum_{v \in f_j} \frac{1}{PRR(u,v)} \leq D \qquad \forall j : 1 \leq j \leq m \tag{7}$$

The inverse of $PRR(u,v)$ in Equation 4 is the average transmission count required for a packet to be successfully received by node v from node u. By multiplying p_u and $\frac{1}{PRR(u,v)}$, we obtain the transmission power consumption for one packet at node u. The objective of Equation 4 is to minimize the total transmission power consumption of all the nodes in the network. Equation 5 is the channel constraint, which confines that each node can only pick a channel from n available channels. Equation 6 confines that all nodes in the same data flow must use the same channel. Equation 7 is the end-to-end delay constraint, which gives the limit of the end-to-end transmission count (including retransmissions at each node) for a packet in each flow. End-to-end transmission count is a commonly used metric to represent end-to-end delay as a higher transmission count leads to a longer end-to-end delay. Note that our minimization problem does not depend on the node duty cycle scheduling, so our work can be integrated with energy-efficient MAC protocols with periodic sleeping for further power savings at the cost of longer communication delays.

4.2 Transmission Delay Analysis

One way to analyze the node transmission delay in a WSN is to use the worst-case scenario, where we can assume that all the links in a neighborhood communicate at the same time, such that the most significant interference and delay are incurred. However, due to the lossy nature of wireless links, real-time communication protocols in WSNs are commonly designed to provide only soft probabilistic real-time guarantees [22] [5]. In addition, the traffic patterns at different sources in many wireless sensor networks, such as surveillance applications [23], are usually independently random and unknown a priori. The chance for all the links in a neighborhood to transmit at exactly the same time is very small. Therefore, it is more meaningful to analyze the average case for WSNs. We modify our problem formulation as:

$$min \sum_{v \in V:(u,v) \in G} p_u \times \frac{1}{PRR_{avg}(u,v)}. \tag{8}$$

Correspondingly, the end-to-end delay constraint in Equation 7 is modified as:

$$\sum_{v \in f_j} \frac{1}{PRR_{avg}(u,v)} \leq D \qquad \forall j : 1 \leq j \leq m \tag{9}$$

where $PRR_{avg}(u,v)$ is the average packet reception ratio at node v when the generated traffic at the sender u follows the random distribution.

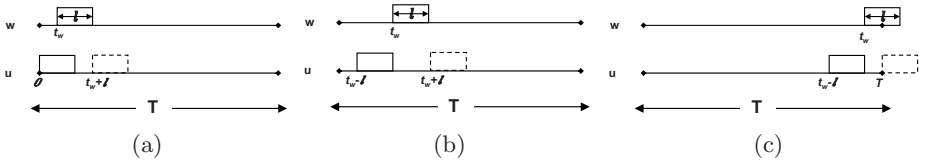


Fig. 4. Probability of packet collision when two nodes have independently random traffic

Note that the probability for more than two nodes to transmit concurrently is small under the random traffic assumption. We assume that at most two nodes in the same interference range may transmit concurrently. We denote the probability that node w's transmission can interfere with node u's transmission as $P(u, w)$ and the packet reception ratio at the receiver v from u's transmission under w's interference as $PRR(u, v, w)$. We can use Equation 10 to estimate the average transmission count for node u to successfully transmit a packet to v when u and w follow the independent random traffic pattern.

$$\frac{1}{PRR_{avg}(u,v)} = (1 - \sum_{w \in I(u)} P(u,w)) \frac{1}{PRR(u,v,v)} + \sum_{w \in I(u)} P(u,w) \times \frac{1}{PRR(u,v,w)} \tag{10}$$

In Equation 10, $PRR(u, v, v)$ is the packet reception ratio at receiver v when there is no interference to sender u's transmission.

Note that $P(u, w)$ in Equation 10 is the probability that node u and node w transmit packets concurrently. To derive $P(u, w)$, we assume that each source node has the same packet rate, 1 packet per T seconds, with a packet length l. We denote the start time of the transmission at node u and w as t_u and t_w, respectively. With the assumptions that the start time of every packet on the source node follows the uniform distribution and each intermediate node forwards packets immediately after receiving, we can calculate $P(u, w)$ as follows:

$$P(u,w) = \begin{cases} \int_0^l \int_0^{t_w+l} \frac{1}{T^2} dt_u dt_w & \text{if } 0 < t_w \leq l; \\ \int_l^{T-l} \int_{t_w-l}^{t_w+l} \frac{1}{T^2} dt_u dt_w & \text{if } l < t_w \leq T-l; \\ \int_{T-l}^{T} \int_{t_w-l}^{T} \frac{1}{T^2} dt_u dt_w & \text{if } T-l < t_w \leq T. \end{cases} \tag{11}$$

Figure 4 illustrates the three cases in Equation 11. In the first case, when $t_w \leq l$, collision happens under the condition that $t_u \leq t_w + l$. In the second case, when $t_w \in (l, T-l]$, collision happens under the condition that $t_u \in (t_w - l, t_w + l)$. In the third case, when $t_w \in (T-l, T]$, collision happens only when $t_u \in (t_w - l, T]$. Note that the independently random traffic pattern assumption can be relaxed in the average PRR estimation. When the traffic pattern is not random, we can use empirical on-line testing to find the collision probability.

By integrating the three cases in Equation 11, we get the collision probability between two nodes in one period T as:

$$P(u,w) = \frac{l(2T-l)}{T^2} \tag{12}$$

Based on the models we established in Section 3, given a power level for each node in the network and a channel assignment to the data flows, we can compute the PRR for each receiving node under the interference from another node. By using Equation 10, we can derive the average transmission count for every node and further calculate the end-to-end delay of every flow, as well as the total system power consumption of the network for the given combination of power levels and channels. Our optimization objective is to find the combination with the least power consumption while the delay of every data flow is shorter than the given constraint.

5 Algorithm Design

The problem formulated in Section 4 is a complex combinatorial optimization problem with a huge search space. Suppose there are j nodes forming m flows in the network. The total available number of channels on the equipment is n. Each mote can use k different power levels to transmit. The combinatorial search space has a size of $n^m \times k^j$. Therefore, we propose to use *Simulated Annealing (SA)* [24], a well-known meta-heuristic, to solve this problem. SA is commonly used to find suboptimal solutions when the search space is huge and discrete, which makes SA well suited for our problem because all possible configuration states are discrete, as the selection of channels and power levels are discrete numbers. Note that although the original SA algorithm is centralized, SA can be extended to run in a distributed way with slightly worse performance [25]. Therefore, our solution can also be extended to run on the sensor nodes in the network in a distributed way. The detailed extension is beyond the scope of this paper. In addition, please note that many real-world WSN applications adopt many-to-one communication [26] [27] for data collection, in which the sink is usually a sensor mote connected to the base station, such as a computer. The base station is commonly used to make centralized decisions for these applications.

Simulated Annealing is a probabilistic method for optimization problems. It transposes the process of the annealing of metal, in which the temperature of the metal is gradually decreased, to the solution search of the optimization problem. In each step, the algorithm considers some neighbor states of the current state, and chooses a valid neighbor state for the next state according to a probabilistic function established on the optimization goal. Two major parts of *SA* are the neighbor state generation and the transition probability. The neighbor state generation scheme requires that every two adjacent states have a short distance. The transition probability is to decide whether the system should go to the next state, *i.e.*, the neighbor state generated in the neighbor generation part.

The objective of our problem is to minimize the total transmission power consumption for the network under an end-to-end delay constraint. The configuration space consists of all the channel assignment and power configuration combinations. Based on a given channel and power configuration, the system proceeds to the next configuration by performing an elementary modification. The elementary modification is defined as a channel change on one of the flows or a power level change on one of the nodes. The pseudo code of our algorithm is given in Algorithm 1.

The algorithm starts with an initial "temperature" T_{ini} and an initial configuration C_{ini} with an initial power consumption P_{ini}. It then looks for a neighbor configuration as the next configuration state, C_{temp}. After a neighbor is found, the algorithm first checks if the delay $delay_i$ of every data flow under the neighbor configuration meets the delay constraint D. If the constraint is met, the algorithm calculates the power consumption difference, ΔP, at the neighbor state and the current state. However, if the constraint is violated, the algorithm adds a *Penalty* to ΔP. The *Penalty* is a parameter that needs to be tuned for the experiment in order to get a good solution. It helps the algorithm to avoid being trapped at a local minimum. The algorithm then checks if the power consumption is reduced. If the power consumption is reduced, the neighbor configuration is accepted. However, if the neighbor configuration causes an increased ΔP for power consumption, the algorithm calculates a probability by the exponential expression $e^{-\frac{\Delta P}{T}}$ and accepts the neighbor configuration based on this probability. After each iteration, the "temperature" is decreased by a factor of ρ. The algorithm ends when the "temperature" is smaller than the threshold T_{end}.

Algorithm 1. Simulated Annealing for Power Consumption Minimization

Denote delay constraint as D, the stop flag as T_{end}, and the starting flag as T_{ini}. The initial channel configuration is C_{ini}. P_{ini} is the initial power consumption. ρ is the factor of temperature decreasing.
$T \Leftarrow T_{ini}$, $C \Leftarrow C_{ini}$,$n \Leftarrow 0$, $P \Leftarrow P_{ini}$
while $T \geq T_{end}$ **do**
  Find neighbor configuration C_{temp}. Calculate power consumption P_{temp} and $delay_i$ for each data flow i
  if $\forall delay_i \leq D$ **then**
    $\Delta P \Leftarrow P_{temp} - P$;
  else
    $\Delta P \Leftarrow P_{temp} + Penalty - P$;
  end if
  if $\Delta P \leq 0$ **then**
    $C \Leftarrow C_{temp}$; $P \Leftarrow P_{temp}$
  else
    if $e^{-\frac{\Delta P}{T}} \geq random()$ **then**
      $C \Leftarrow C_{temp}$; $P \Leftarrow P_{temp}$
    end if
  end if
  $n \Leftarrow n + 1$, $T \Leftarrow \rho^n T_{ini}$
end while

6 Empirical Results

In this section, we present the evaluation results of our configuration algorithm on a hardware testbed.

6.1 Testbed Setup and Baselines

Our testbed consists of 25 Tmote motes. Two different topologies used for the experiments are shown in Figure 5. Node 13, as the base station, consists of 5 real motes in the experiment, which emulates a super node with 5 radios. Independent uniform random traffic generator are implemented on each source node.

In the RSS measurement phase, every mote in the network takes turn to act as the sender and broadcasts packets using different power levels on different channels. While the sender is sending packets at a certain power level on a fixed channel c, all other nodes, acting as listeners, iterate through channel $c-1$, c and $c+1$, and record the received signal strength on each channel. The reason we choose three channels to listen is because only the same channel and adjacent channels show the approximate linear PRR-Power pattern, as discussed in Section 3. In the experiments, we choose 5 discrete power levels: 3, 10, 17, 24 and 31, as the transmission power for the model establishment. This helps us to reduce the solution search space and speed up the experiments. After collecting all the RSS measurements, we import the data to the Simulated Annealing optimization program we implemented in MATLAB to compute the channel and power configuration.

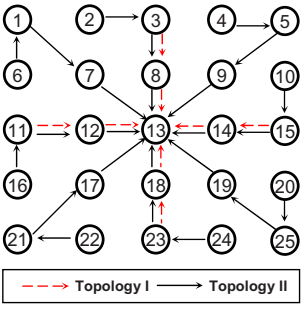


Fig. 5. Topologies used in experiments

We choose the following two baselines for comparison. The first baseline, called *Orthogonal*, uses only orthogonal channels for channel assignment and computes the power configuration by Simulated Annealing. The second baseline, called *Random*, also uses Simulated Annealing to find the desired power level for each node, but randomly assigns overlapping channels to flows. We use two metrics to evaluate the performance of these three protocols. The first metric is *average end-to-end transmission count*, which evaluates the end-to-end delay performance. The second metric is *transmission power consumption per packet*, which is the ratio between total transmission power and the number of packets transmitted. This metric evaluates the energy efficiency performance.

6.2 Different Delay Constraints

We first evaluate the three schemes under different transmission count constraints. In this experiment, we use Topology I in Figure 5 with nine motes,

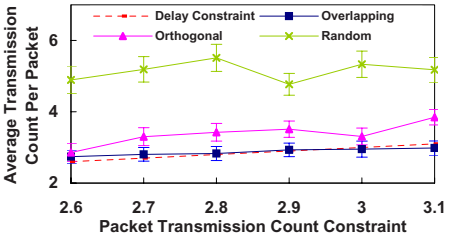


Fig. 6. Delay under different end-to-end transmission delay constraints

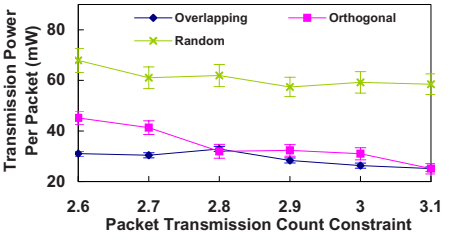


Fig. 7. Power consumption under different end-to-end transmission delay constraints

forming 4 flows. Three channels, 16, 17 and 18, are used. Channels 16 and 18 are orthogonal channels while channel 17 overlaps with channels 16 and 18. Figure 6 shows the average end-to-end delay under different constraints. The overlapping scheme achieves a smaller average end-to-end transmission count than the two baseline schemes. In addition, the delay of our overlapping scheme is closest to the constraints. The reason for the superior performance of our overlapping scheme is that it takes advantage of overlapping channels with carefully selected power and channel configuration by the Simulated Annealing algorithm to reach suboptimal solutions. With more channel resources to use, the overlapping scheme achieves a better configuration solution than the other two protocols. When the constraint becomes looser, all the schemes yield higher end-to-end transmission counts. The results demonstrate that the end-to-end delay in the network is adaptive to the change of the delay constraint.

Figure 7 shows the transmission power consumption per packet for different constraints. Among all three schemes, the overlapping scheme consumes the least transmission power. This is because the overlapping scheme utilizes all the available channel resources and carefully chooses the most appropriate transmission power to reduce the interference among nodes such that the power consumed by retransmissions is significantly reduced. When the constraint is greater than 2.8, the performance of Orthogonal is close to that of our scheme. However, when the constraint is tight, Orthogonal performs significantly worse than the overlapping scheme. All the three schemes show decreasing trends for power consumption when the end-to-end transmission constraint becomes looser. This is because when the constraint is looser, we have a larger search space for the SA algorithm, likely resulting in a better power configuration.

6.3 Different Flow Numbers

Figures 8 and 9 show the performance of the network with different numbers of data flows. It is important to evaluate the performance of the network under different numbers of flows because multiple flows may need to share channels when the number of flows increases. In these experiments, we use Topology II

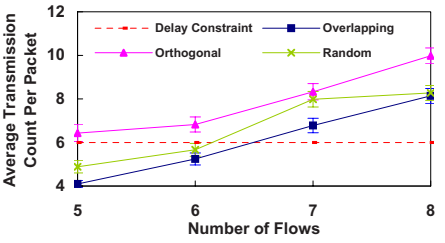


Fig. 8. Delay under different numbers of data flows

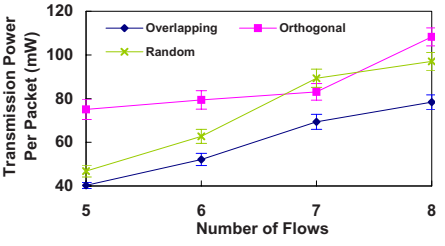


Fig. 9. Power consumption under different numbers of data flows

in Figure 5, where 25 motes are organized as a 5 by 5 grid. The base station is placed in the center, similar to the previous two experiments. Each data flow has three hops and we gradually increase the number of data flows in the network from 5 to 8. We use 5 overlapping channels, from channel 16 to channel 20, where 3 channels are orthogonal.

Figure 8 shows that the average transmission count per packet increases when the number of data flows increases. This is because more data flows cause more interferences in the network and more flows need to share the same channels for data transmissions, which results in more intra-channel interferences and competition. The same trend can be observed in Figure 9 for power consumption. Among all the three schemes, the overlapping scheme performs best for both the average transmission count and average power consumption. This is because the overlapping scheme utilizes most channel resources to reduce the reuse of each channel, which leads to less intra-channel interference. In the meantime, the overlapping scheme also carefully configures the transmission power to reduce the interferences among adjacent channels. Note that Orthogonal performs the worst because a greater number of flows need to share channels when there are only 3 orthogonal channels available. The increased channel sharing leads to a higher degree of channel competition and intra-channel interferences, and thus more packet retransmissions.

7 Conclusions

In this paper, we have conducted empirical studies to investigate the interferences among overlapping channels. Our results show that overlapping channels can also be utilized for improved real-time performance if the transmission power is carefully configured. In order to minimize the overall power consumption of a network with multiple data flows under end-to-end delay constraints, we formulate a constrained optimization problem to configure the transmission power level for every node and assign overlapping channels to different data flows. Since the optimization problem has an exponential computational complexity, we then present a heuristic algorithm designed based on Simulated Annealing to find a suboptimal solution. Our extensive empirical results on a 25-mote testbed

demonstrate that our algorithm reduces both the end-to-end communication delay and overall transmission power consumption, compared with two baselines: a scheme using only orthogonal channels and a scheme using simple policy to assign overlapping channels.

References

1. Cc2420 2.4 ghz ieee 802.15.4 / zigbee-ready rf transceiver, http://www.chipcon.com
2. Gupta, P., Kumar, P.R.: The capacity of wireless networks. IEEE Transactions on on Information Theory 46(2) (2000)
3. Zhang, J., Zhou, G., Huang, C., Son, S.H., Stankovic, J.A.: TMMAC: An energy efficient multi-channel mac protocol for ad hoc networks. In: ICC (2007)
4. Zhou, G., Huang, C., et al.: MMSN: Multi-frequency media access control for wireless sensor networks. In: INFOCOM (April 2006)
5. Wang, X., Wang, X., Fu, X., Xing, G., Jha, N.: Flow-based real-time communication in multi-channel wireless sensor networks. In: Roedig, U., Sreenan, C.J. (eds.) EWSN 2009. LNCS, vol. 5432, pp. 33–52. Springer, Heidelberg (2009)
6. Wu, Y., Stankovic, J., He, T., Lin, S.: Realistic and efficient multi-channel communications in dense sensor networks. In: INFOCOM (2008)
7. Liu, H., Yu, H., Liu, X., Chuah, C.-N., Mohapatra, P.: Scheduling multiple partially overlapped channels in wireless mesh networks. In: ICC (2007)
8. Feng, Z., Yang, Y.: Scheduling multiple partially overlapped channels in wireless mesh networks. In: WCNC (2008)
9. Rad, A.H.M., Wong, V.W.: Partially overlapped channel assignment for multi-channel wireless mesh networks. In: ICC (2007)
10. Sha, M., Xing, G., Zhou, G., Liu, S., Wang, X.: C-mac: Model-driven concurrent medium access control for wireless sensor networks. In: INFOCOM (2008)
11. Demirbas, M., Song, Y.: An rssi-based scheme for sybil attack detection in wireless sensor networks. In: WoWMoM (2006)
12. Stankovic, J.A., Abdelzaher, T., Lu, C., Sha, L., Hou, J.: Real-time communication and coordination in embedded sensor networks. Proceedings of the IEEE 91(7) (2003)
13. Caccamo, M., Zhang, L.Y., Sha, L.: An implicit prioritized access protocol for wireless sensor networks. In: RTSS (2002)
14. He, T., Stankovic, J., Lu, C., Abdelzaher, T.: SPEED: A stateless protocol for real-time communication in sensor networks. In: ICDCS (2003)
15. Xing, G., Sha, M., Huang, J., Zhou, G., Wang, X., Liu, S.: Multi-channel interference measurement and modeling in low-power wireless networks. In: RTSS (2009)
16. Talzi, I., Hasler, A., Gruber, S., Tschudin, C.: Permasense: investigating permafrost with a wsn in the swiss alps. In: EmNets (2007)
17. Goldsmith, A.: Wireless Communications. Cambridge University Press, Cambridge (2005)
18. Karenos, K., Kalogeraki, V.: Real-time traffic management in sensor networks. In: RTSS (2006)
19. Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., Anderson, J.: Wireless sensor networks for habitat monitoring. In: ACM Workshop on Sensor Networks and Applications (2002)
20. Maimour, M.: Maximally radio-disjoint multipath routing for wireless multimedia sensor networks. In: WMuNep (2008)

21. Deng, J., Han, R., Mishra, S.: Decorrelating wireless sensor network traffic to inhibit traffic analysis attacks. Elsevier Pervasive and Mobile Computing Journal, Special Issue on Security in Wireless Mobile Computing Systems (2006)
22. Chipara, O., He, Z., Xing, G., Chen, Q., Wang, X., Lu, C., Stankovic, J., Abdelzaher, T.: Real-time power-aware routing in sensor networks. In: IWQoS (2006)
23. He, T., Vicaire, P., Yan, T., Cao, Q., Zhou, G., Gu, L., Luo, L., Stoleru, R., Stankovic, J.A., Abdelzaher, T.F.: Achieving long-term surveillance in vigilnet. In: INFOCOM (April 2006)
24. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
25. Nabhan, T.M., Zomaya, A.Y.: A parallel simulated annealing algorithm with low communication overhead. IEEE Trans. Parallel Distrib. Syst. (1995)
26. Selavo, L., Wood, A.D., Cao, Q., Sookoor, T., Liu, H., Srinivasan, A., Wu, Y., Kang, W., Stankovic, J.A., Young, D., Porter, J.: Luster: wireless sensor network for environmental research. In: SenSys (2007)
27. Jeong, J., Culler, D.E., Oh, J.-H.: Empirical analysis of transmission power control algorithms for wireless sensor networks. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2005-16 (November 2005)

Privacy-Preserving Reconstruction of Multidimensional Data Maps in Vehicular Participatory Sensing

Nam Pham[1], Raghu K. Ganti[1], Yusuf S. Uddin[1], Suman Nath[2], and Tarek Abdelzaher[1]

[1] University of Illinois at Urbana-Champaign
{nampham2,rganti2,mduddin2,zaher}@illinois.edu
[2] Microsoft Research
sumann@microsoft.com

Abstract. The proliferation of sensors in devices of frequent use, such as mobile phones, offers unprecedented opportunities for forming self-selected communities around shared sensory data pools that enable community specific applications of mutual interest. Such applications have recently been termed *participatory sensing*. An important category of participatory sensing applications is one that construct maps of different phenomena (e.g., traffic speed, pollution) using vehicular participatory sensing. An example is sharing data from GPS-enabled cell-phones to map traffic or noise patterns. Concerns with data privacy are a key impediment to the proliferation of such applications. This paper presents theoretical foundations, a system implementation, and an experimental evaluation of a perturbation-based mechanism for ensuring privacy of location-tagged participatory sensing data while allowing correct reconstruction of community statistics of interest (computed from shared perturbed data). The system is applied to construct accurate traffic speed maps in a small campus town from shared GPS data of participating vehicles, where the individual vehicles are allowed to "lie" about their actual location and speed at all times. An extensive evaluation demonstrates the efficacy of the approach in concealing multi-dimensional, correlated, time-series data while allowing for accurate reconstruction of spatial statistics.

1 Introduction

An emerging category of applications focus on collecting and sharing sensor data for the purpose of characterizing aggregate real-world properties, such as computing community-wide statistics or mapping physical phenomena of common interest. These applications are termed *participatory* sensing applications [1]. Examples of these applications include vehicular sensor networks for collecting and sharing traffic data [2], bicycle networks to collect and share bikers' paths [3], and cell phone based buddy networks to collect and share location and activity information [4]. An important category of participatory sensing applications

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 114–130, 2010.

is one where users share location-tagged data to construct maps of different phenomena (e.g., traffic speed, pothole, pollution).

One main problem in participatory sensing applications that share location-tagged data is privacy. For example, a community of environmentalists might want to collectively measure pollution on city streets and share that information to construct city-scale pollution maps. Since such data are location-tagged, a key question is to enable correct geographic mapping without revealing private location information of individuals collecting the location-sensitive data. The problem becomes non-trivial in the absence of a shared trusted entity that can be used to sanitize the data. Moreover, since the data itself, such as GPS traces, may reveal user identity, anonymity is not the answer to the privacy problem.

To address the above challenge, in this paper, we solve the privacy problem via data perturbation. Perturbing data on the client-side prior to sharing empowers clients by giving them the freedom to "lie" about both their data and the context (such as location) where it was collected. Clients share their perturbed data with an entity we call the *aggregation server*. It is responsible for computing the aggregate statistics of interest. Clients trust the server with computing the statistics but do not want to reveal their private data to it for privacy reasons. When receiving perturbed data, in addition to computing the community statistics, the server may try to guess the original individual user data, which we call a *privacy attack* . This paper designs perturbation algorithms that protect against privacy attacks, while ensuring accurate reconstruction of community statistics. The contribution lies in solving the above problem for the case of multidimensional correlated time-series data (such as correlated sensor data streams).

From an algorithmic perspective, the fundamental limitation of previous approaches is that they do not consider privacy-preserving perturbation and reconstruction when each user shares *multiple correlated* private data streams. For example, when collecting speed at different locations to build a city speed map, both speed and location are private since a client might not want to admit, say, to speeding, and might not want their location to be tracked.

We provide a solution to the general problem of ensuring privacy for multi-stream data of individuals while allowing community statistics to be reconstructed accurately. We develop a correlated noise model that can be utilized for perturbing location-tagged data in a way that protects both data and location privacy. We evaluate the approach using a traffic monitoring application implemented using an existing architecture called PoolView [5]. The application follows a client-server model. The client-side software collects data from the client's GPS device, perturbs the data and shares those with an aggregation server. The aggregation server then estimates useful community statistics from perturbed data and makes those statistics available for community access. Empirical measurements show that the approach results in accurate reconstruction of speed maps from perturbed data while preventing the reconstruction of individual client data and location information.

The rest of this paper is organized as follows. We first develop the reconstruction algorithm of the joint probability distribution in Section 2. Privacy properties are discussed in Section 3. Section 4 and Section 5 describe simulation-based evaluation and deployment-based evaluation, respectively. Finally, Section 6 concludes the paper.

2 Joint Probability Density Function Reconstruction

The main contribution of this paper lies in the algorithm to accurately reconstruct the community joint density given the perturbed multidimensional stream data and the noise density information. Any statistical question about the community can be answered using the reconstructed joint density. There have been many efforts on the community distribution reconstruction. Agrawal et al. [6] proposed a Bayesian-based reconstruction of the probability distribution. In [7], the authors use the Expectation Maximization (EM) algorithm to estimate one-dimensional distribution from data perturbed with Gaussian noise. In our previous work [5], we employed the Tikhonov-Miller deconvolution technique to estimate the community distribution. However, all of these algorithms are developed to reconstruct a one-dimensional distribution. Hence, they do not scale to the problem of multidimensional distribution reconstruction. In this section, we present an iterative algorithm to estimate the discretized joint distribution of multidimensional data streams.

Let the number of data streams that each user wants to share be M. The shared data from each user are assumed to be drawn from a multivariate random variable $X = (X_1, X_2, \ldots, X_M)$, thus each data point is a length M vector. The reconstruction algorithm does not distinguish which data points are from which user. Therefore, we can define the set of all data points from all users as $\bar{X} = \{x_1, x_2, \ldots, x_n\}$ where x_i is a length M data point, and n is the total number of data points from all users.

Each data point is perturbed by adding an M-dimensional noise data point generated from a known joint distribution $f_N(N_1, N_2, \ldots, N_M)$ which is known to all participating users (or rather to their client-side software). An aggregation server receives the set of n perturbed data points from all users denoted as $\bar{Y} = \{y_1, y_2, \ldots, y_n\}$. We want to estimate the joint distribution of X which is $f_X(X_1, X_2, \ldots, X_M)$ given the shared data $\bar{Y}$ and the knowledge of the noise distribution f_N.

Let us denote the sample space of X_i as Ω_i. Thus, the sample space of X is $\Omega = \Omega_1 \times \Omega_2 \times \ldots \times \Omega_M$. In order to reconstruct the density of X, we first discretize the the sample space Ω. The sample space of X_i is partitioned into K_i bins (may not be uniform) denoted as $\{\Omega_i^1, \Omega_i^2, \ldots, \Omega_i^{K_i}\}$. Thus Ω containes $K = K_1 \times K_2 \times \ldots \times K_M$ M-dimensional bins in which the value of the density function is constant. The more the number of bins, the better the discrete density approximates the continuous density. To simplify the notation, the following symbols are introduced:

- ω_I: the I^{th} bin of Ω, thus $\Omega = \cup_{\omega_I} \omega_I$.
- $\Theta = \{\theta_1, \theta_2, \ldots, \theta_K\}$: where $\theta_i = f_X(X)$ with $X \in \omega_I$, is the set of all density parameters to be estimated.
- m_{ω_I}: the volume of ω_I, a proper discrete density parameters Θ should satisfy

$$\sum_{\omega_I} \theta_I m_{\omega_I} = 1 \tag{1}$$

To estimate Θ, our approach is to employ the maximum likelihood framework. We need to find the density function parameters which maximize the log likelihood of the data $\bar{X}$ given the observations $\bar{Y}$

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmax}} \log f_{X;\Theta}(\bar{X}|\bar{Y}) \tag{2}$$

The notation $f_{X;\Theta}$ means that the likelihood of X is computed using the discrete distribution Θ. Unfortunately, the likelihood can not be computed directly at the aggregation server because only $\bar{Y}$ is known while $\bar{X}$ is missing. A common procedure to solve the maximum likelihood estimation with incomplete information is the EM algorithm [8]. To use the EM algorithm, the following auxiliary function $Q(\Theta|\hat{\Theta}^k)$ is defined:

$$Q(\Theta|\hat{\Theta}^k) = E_{X|Y}\left[\log f_{X;\Theta}(\bar{X})|\bar{Y}, \hat{\Theta}^k\right] \tag{3}$$

The auxiliary function Q is actually the expectation of the likelihood in (2) with respect to X using the density of X computed from the previous step which is $\hat{\Theta}^k$. The EM algorithm consists of two steps:

- E-step : Given the density computed from the k^{th} step, compute the value of $Q(\Theta|\hat{\Theta}^k)$
- M-step : Compute $\hat{\Theta}^{k+1} = \operatorname{argmax}_{\Theta} Q(\Theta, \hat{\Theta}^k)$

Next, we will derive a closed form expression for Q, the optimal solution which maximizes the likelihood function and analyze the convergence of the algorithm.

Theorem 1. *(E-step) The value of $Q(\Theta|\hat{\Theta}^k)$ is given by:*

$$Q(\Theta|\hat{\Theta}^k) = \sum_{\omega_I} \hat{\theta}^k_{\omega_I} \log(\theta_{\omega_I}) \phi^k_{\omega_I} \tag{4}$$

Where

$$\phi^k_{\omega_I} = \frac{1}{N} \sum_{j=1}^{N} \frac{f_N(y_j - \omega_I)}{f^k_{Y;\hat{\Theta}^k}(y_j)} \tag{5}$$

$$f_{Y;\hat{\Theta}^k}(y_j) = \sum_{\omega_I} f_N(y_j - \omega_I) \hat{\theta}^k_{\omega_I} \tag{6}$$

$$f_N(y_j - \omega_I) = \int_{\omega_I} f_N(y_j - \gamma) d\gamma \tag{7}$$

Proof. See Appendix A.1.

Theorem 2. *(M-step) The value of* $\hat{\Theta}^{k+1}$ *maximizing the auxiliary function* $Q(\Theta|\hat{\Theta}^k)$ *is given by*

$$\hat{\theta}_{\omega_I}^{k+1} = \frac{\phi_{\omega_I}^k}{m_{\omega_I}} \hat{\theta}_{\omega_I}^k \tag{8}$$

Proof. See Appendix A.2.

In the next theorem, we show that the EM algorithm for this problem is guaranteed to converge to the maximum likelihood solution which is the solution for (2). Therefore the likelihood value increases slowly as it approaches the optimal solution. Thus a stopping condition for the algorithm is when the likelihood difference between two consecutive steps is sufficiently small.

Theorem 3. *The estimated density function given by the algorithm converges to the maximum likelihood solution* $\hat{\Theta}$ *defined in the Equation (2).*

Proof. We will first prove that $Q(\Theta|\hat{\Theta}^k)$ is concave in θ_{ω_I}. In Theorem 1, we prove that the value of the auxiliary function $Q(\Theta|\hat{\Theta}^k) = \sum_{\omega_I} \hat{\theta}_{\omega_I}^k \log(\theta_{\omega_I}) \phi_{\omega_I}^k$ which is the non-negative linear combination of $\log(\theta_{\omega_I})$. Since $\log(x)$ is a concave in x, the non-negative linear combination of $\log(x)$ functions is also concave. Thus Q is concave in θ_{ω_I}.

Wu et al. [9] showed that the value of the likelihood increases after each iteration. Because Q is concave, the iterative algorithm will finally converge to $\hat{\Theta}$ which maximizes the likelihood function defined in (2).

3 Perturbation of Location and Data

Having presented a general algorithm for reconstruction of community statistics, it remains to decide on the perturbation function. This question is equivalent to choosing the noise probability density function, from which noise samples are chosen. Perturbation is application specific, since it depends on what is being perturbed. We consider the class of applications where we perturb location-tagged data collected by vehicles.

In our application, individuals collect GPS longitude, GPS latitude, speed and (coarsely discretized) time, using their own GPS devices. Once the aggregation server receives perturbed data from participants, the community joint density (i.e., the joint density of longitude, latitude and speed) is reconstructed using the above reconstruction algorithm. Speed-related statistics are then computed as a function of location on the map from the reconstructed joint density. In this paper, we present useful community statistics that can be computed from the estimated multidimensional density such as community average speed, speed distribution, car density, and percentage of speeding vehicles on different streets.

The application was deployed on top of our existing architecture for participatory sensing called PoolView [5]. PoolView is a generic client-server based architecture that enables individuals to collect, archive, and share sensor data with a community On the client side, PoolView provides software that collects

sensor data from specific devices (e.g., Garmin GPS). We modified the PoolView client to use our new multidimensional data perturbation scheme. On the server side, we implemented the multidimensional density reconstruction algorithm and the algorithms used to estimate the aforementioned statistics.

3.1 The Perturbation Model

In this section, we propose an algorithm that generates fake (but realistic-looking) vehicle traces that perturb true user location and speed in a way that protects them from being estimated. The vehicle traces are recorded as displacements from an origin (of a coordinate framework) that lies at some agreed upon point in the city in question. These displacements, which we henceforth call *perturbation traces*, will then be added to real routes to generate perturbed routes. There has been many research efforts on generating vehicle traces in prior work [10,11,12,13]. We can utilize one of those models to generate perturbation traces for our application. However, the vehicle traces used for perturbation do not need that level of accuracy. Thus, we develop a simplified model that generates perturbation traces using a minimal number of simple parameters.

It is key that the perturbation traces generated resemble real traces for the city in question. For example, in a city with a lot of curvy roads, generated perturbation traces containing only straight segments will not help conceal the identifying characteristics of the roads actually traveled. A robust perturbation trace generation algorithm must therefore incorporate as many features of the actual map as possible.

Our perturbation trace generation algorithm generates traffic routes made of sequences of straight line segments, each of a length drawn from the distribution of the lengths of city blocks. These segments are at angles generated from the distribution of city street intersection angles. This distribution heavily favors 0 degree angles (continuing forward past an intersection) and 90 degree turns. Other angles are generated with lower probability. We ignore U-turns because they occur with a very small probability. For speed, we use a sine curve for each road segment that peaks in the middle of the segment and slows down towards the beginning and end. The peak is drawn from the distribution of city street speed limits. The slowest point is a uniformly-distributed random fraction of the peak. These traces represent displacement to actual routes. This displacement can be scaled to control the noise variance.

Finally, for the purpose of reconstructing the community joint distribution, we need the joint distribution of the generated perturbation trace (the noise). Since it is hard to come up with an analytic solution for the joint distribution of the noise, we generate this distribution numerically. First, we generate a pool of noise data points from the model then *a non-parametric density estimation with smoothing* [14] is employed to estimate the joint distribution. In this application, 5000 vehicle traces, each of which contains 40 data points, are generated and used as input to the density estimation algorithm, which generates the joint distribution.

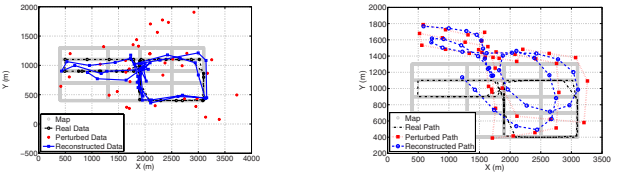
3.2 Achieved Privacy

In this section, we analyze the extent of privacy offered to *individual* user data using our perturbation scheme. The information available to the aggregation server includes the perturbed data, the noise density function (known by the server) and the map on which the user traveled. First, note that the reconstruction algorithm proposed in this paper can not be used to reconstruct individual's real data from those information. Our proposed algorithm can only reconstruct community distribution from shared data of a reasonable number of participants. Using the available information, the malicious server can employ filtering techniques to remove additive noise from the perturbed data. We call this kind of attack *filtering attack*.

In this paper, we analyze a filtering attack which applies a Wiener filter to remove additive noise from perturbed data. The Wiener filter uses the noise density information to filter the noise from perturbed data. One important assumption that the Wiener filter makes is the noise samples are independent. However, this assumption fails because the noise samples generated by our algorithm are correlated which makes the estimated data traces follow the perturbed path instead of real path. For demonstration, we perturb a real user location trace with both correlated noise generated by our algorithm and independent Gaussian white noise and then perform the Wiener filter on both perturbed data set.

The result of the Wiener attack in the case of Gaussian white noise is shown in Figure 1(a). The reconstructed path is very close to the real path and the reconstruction error is less than one block which means that the attacker can easily figure out the place where the user have been. Figure 1(b) shows the real path, perturbed path and the reconstructed path for the perturbation technique we developed in this paper. We see that the reconstructed path follows the perturbed path. Therefore, the Wiener filter attack does not work as desired for the attacker. Users might want to increase the variance of the generated noise to get more privacy, but the reconstruction error might increase as well. Therefore, it is important to balance the trade off between privacy and accuracy.

The second type of attack considered in this paper is the range attack. It is possible to conduct the range attack in applications where the ranges of both



(a) Reconstruction of user location with Gaussian noise

(b) Reconstruction of user location with correlated noise

Fig. 1. Reconstruction of user location perturbed with different noise model

the real data and the generated noise are finite. In this case, real data values can be inferred if boundary values of the perturbed data are observed. For example, suppose the real speed of a vehicle is in the range [0 to 50] and the generated noise is also in the range [0 to 50]. If the perturbed speed is 100, the attacker knows with certainty that the true speed is 50. In general, if the perturbed values are close to the boundary, privacy can be violated. In applications involving GPS location as a private variable, however, this attack is not effective. GPS location refers to a point of the globe. Perturbing that location by a few miles is sufficient for privacy, yet the perturbed location still refers to a point on the globe. In other words, the perturbed coordinates always refer to a valid data point. An exception is when map information is used to infer noise. For example, at coastal areas, one may safely assume that vehicles do not move on water, which generates a boundary on valid locations. The map-based attack will be discussed shortly. In general, the effect of range-based attacks can be mitigated if the noise distribution has a long tail such that arbitrarily large values are allowed with an arbitrarily low probability. (Many distributions, including Gaussian, have this property.) In this case, the range is infinite. There is no maximum value for the perturbed signal that can be used to breach privacy.

Another popular type of attack against additive-noise perturbation techniques is the *leak attack* [15]. In this type of attack, the attacker may be able to estimate the seed of the pseudo random number generator which generates the noise curve if he can guess a few true data values. Then this seed can be used to generate the noise curve used by the user since the noise distribution is known. However, with our perturbation scheme, this attack is not possible because we only use the random number generator to generate the model parameters (e.g., number of turns, speed of each segment). The additive noise is then generated using those parameters and the model developed earlier in this section.

A vulnerability of our perturbation scheme is that it is possible to combine the real map with a clever estimation technique to estimate the most likely traveled path. We call this attack scheme a *map-based attack*. At this moment, it is still unknown if there exists a good map-based attack against our perturbation scheme. In this paper, we argue that finding an efficient map-based attack is hard. One possible way to conduct the map-based attack is to look at the sequence of the turning angles in the GPS trajectory data. Since the probability that the noise angle and the real angle cancel out is pretty small, the turning angles from the perturbed data contain some information about the real turning angles. Combining with the map, it is possible to find the most probable traveled path. It is not easy, however, to find the likelihood of the real turning angle given the perturbed path. Because the perturbed path is created by adding the coordinates of the real path and the noise path, the angle in the perturbed path is not only depend on the angle of both real path and noise path but also depend on the magnitudes of those. In the upcoming sections, we only evaluate the immunity of our perturbation scheme against filtering attacks.

4 Simulation Results

In this section, we evaluate the performance of the traffic mapping application with simulated data. The advantage of using simulated data is to give total control over traffic parameters, (e.g., average community speed, speed map), which is hard to accurately measure in a real application. In addition, vehicular traces can be generated for a large numbers of "virtual" users makes it possible to evaluate the accuracy of the reconstruction algorithms. We also evaluate the accuracy computation of the community average speed using the reconstructed density in this section.

We use the ONE (Opportunistic Network Environment) [16] simulator to generate artificial traces of vehicle movements in a small city setup. The map used in this simulation is a part of Helsinki city and is distributed with the ONE simulator. The simulator supports Map Based Movement models that can import map data and constrain vehicle movement to the streets and roads of the imported map.

Our goal is to make the data get out from the simulator as realistic as possible. The input map for the simulator is extracted from a real map and is shown in Figure 2 with the X and Y coordinates ranging from 0 to 4000 meters and 0 to 3600 meters respectively. Vehicle speeds are chosen to be Gaussian with mean 30mph and standard deviation of 10mph. Trip data, including X and Y coordinates and vehicle speed, are sampled at a frequency of 1 Hz, and are stored in an external file for later use. The simulated data are then perturbed with perturbation traces generated by the algorithm discussed in Section 3.1. The perturbed data are then submitted to the aggregation server.

Fig. 2. The map used in simulation

We collect data from 120 users, each of which contains 80 data points, from the simulation. In order to reconstruct the community joint distribution, we first have to specify the range of each dimension and the number of bins in each dimension. Those parameters are summarized in Table 1. In this simulation, we discretize the location in 100mx100m bins which is small enough to capture the street information. For more accurate reconstruction of the joint density, more bins in each dimension might be needed but it would require more user data points and computational time. In this specific traffic application, we are

only interested in the density values corresponding to the street locations. Our proposed algorithm allows us to do the reconstruction on those bins only thus siginificantly reduce the time complexity of the algorithm.

Table 1. Parameters for the reconstruction

Parameter	range of X	range of Y	range of V
Value	0 - 4000 (m)	0 - 3600 (m)	0 - 60 (mph)
Parameter	X bins	Y bins	V bins
Value	40	36	60

Table 2. Noise variance in each data set

Parameter	stddev of X (m)	stddev of Y (m)	stddev of V (mph)
Dataset 1	100	100	4
Dataset 2	500	500	36
Dataset 3	900	900	60
Dataset 4	1500	1500	76
Dataset 5	3000	3000	100

In the first experiment, we study the accuracy of the density reconstruction algorithm under various noise variance. The application must achieve high reconstruction accuracy at a reasonably high noise variance level in order to provide sufficient privacy to users. To achieve this goal, we perturbed the simulation data using five different noise variances shown in Table 2.

We define the accuracy of the density reconstruction as a function of the average accuracy of all the bins:

$$r = \frac{1}{K} \sum_{i=1}^{K} \left(1 - \frac{|\theta_i - \hat{\theta}_i|}{\theta_i} \right) \tag{9}$$

In Equation (9), r is the computed accuracy, θ_i is the true discrete density parameter, $\hat{\theta}_i$ is the estimated density parameter. $\hat{\theta}_i$ is obtained by feeding the real density using real user data points to the density estimation algorithm.

The accuracies of the reconstructions as the function of the number of data points and noise variance are shown in Figure 3. The figure shows five different curves corresponding to the five dataset described above. The X axis is the number of data points which varies from 120 points to 1200 points with 120-point increments. In the results, Dataset 1 achieves highest accuracy while Dataset 5 achieves lowest accuracy.

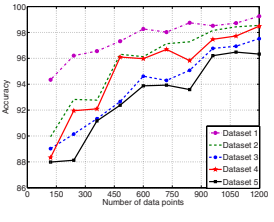


Fig. 3. Percentage reconstruction accuracy as a function of number of data points and noise variance

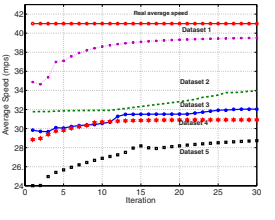


Fig. 4. Community average speed versus number of iterations

Next, we evaluate the achieved privacy for each dataset presented in Table 2. We assume that the attacker uses Wiener filter to estimate vehicle trace of *individuals* from perturbed data and the noise distribution. Beside correlated noise, trip data are also perturbed with Gaussian noises with the same standard deviation for comparision purpose. We perform the estimation on the perturbed vehicle trace of all users and compute the average reconstruction error which is presented in Table 3 below.

Table 3. Reconstruction Error of Individual Data

Dataset	Correlated Noise (m)	Gaussian Noise (m)
Dataset 1	334.5	145.0
Dataset 2	1329.5	153.4
Dataset 3	1942.4	189.8
Dataset 4	3573.6	218.1
Dataset 5	4901.1	223.5

From the Table 3, the reconstruction error for the vehicle traces perturbed with correlated noise is very high as opposed to the Gaussian case in which the error is small. With Dataset 1 (the noise covariance is small) the reconstruction of individual data is still high (about 3 blocks) which means good privacy is achieved. Also, with Dataset 5, although the reconstruction error of individual data is huge (about 40 blocks), the community distribution can still be accurately reconstructed (above 96%).

In the last experiment, we demonstrate the estimation of the community average speed using the joint distribution estimated in the first experiment. In addition, we also want to study the effect of the number of iterations on the accuracy of reconstruction. To compute the community average speed from the community joint distribution $f(X, Y, V)$, we first compute the speed density $f(v)$

$$f(v) = \sum_{x=1}^{40} \sum_{y=1}^{36} f(x, y, v) \Delta_{XY} \tag{10}$$

Equation (10) is the marginalization of the discrete joint density over X and Y dimensions. where $\Delta_{XY} = (4000/40) * (3600/36)$ is the area of a two dimensional bin XY. Then the average speed $\bar{v}$ is computed as $\bar{v} = \sum_{v=1}^{60} v f(v)$.

The result of the experiment is shown in Figure 4. Although Dataset 5 provides users with highest acceptable privacy, the reconstructed average speed is still close to the true value. Another important observation from the graph is that the density reconstruction algorithm requires a very small number of iterations to converge. Results from 5 datasets show that 10 to 15 iterations are sufficient. The accuracy of the algorithm almost does not change after 20 iterations. In the next section, we evaluate the performance of the application using deployment data.

5 Deployment Data

In this section, we evaluate the traffic monitoring application with real deployment data. The data are collected by driving on all the streets within an area shown in Figure 5. There are a total of 15 users, each user drives the streets at will for 10 minutes. During the drive, we use a Garmin Legend [17] GPS device to record location and speed information. The sampling frequency of the device is 15Hz which is enough to record changes in the location and speed since the speed limit in the area is 25 mph.

Fig. 5. Map used to collect data

At the aggregation server side, to do the reconstruction, we need to specify the reconstructed region and the number of bins in each region. The reconstruction parameters are summarized in Table 4. For location, we divide each axis into 30 bins, the width of each bin is 0.01 mile, which is about the width of a street. This is important because, we want to estimate the speed down to the resolution of a street. This can be done by looking at the specific bins corresponding to the target street.

Table 4. Parameters for the reconstruction

Parameter	range of X) (1/100 mile)	range of Y (1/100 mile)	range of V (mph)
Value	0 - 300	0 - 300	0 - 25
Parameter	X bins	Y bins	V bins
Value	30	30	30

Table 5. Noise standard deviation

Parameter	stddev of X (1/100 mile)	stddev of Y (1/100 mile)	stddev of V (mph)
Dataset 1	45	35	5
Dataset 2	75	75	10
Dataset 3	100	100	15
Dataset 4	150	150	20
Dataset 5	300	300	30

In the first experiment, we study the density reconstruction accuracy as a function of the number of data points used for reconstruction. We want to answer the question of how many data points we need to achieve a desired accuracy. Similar to the case of simulation data, we do the perturbation of the data with five different noise data sets each of which has different variance. The details

of the noise datasets are presented in Table 5. The standard deviation of the noise specified in the table is comparable to multiples of the block length (about 75/100 mile), We run the density reconstruction algorithm multiple times, each time with a different number of data points. The data points are randomly picked from the total pool of data points contributed by all users. The number of data points taken for reconstruction is varied from 100 to 800.

The results of the experiment are shown in Figure 6. From the result, the highest accuracy achieved is about 90% at about 800 datapoints while the lowest accuracy is about 83% at about 160 datapoints. The number of data points needed for a good estimate is thus surprisingly low. This can be explained by the observation that since the data points are uniformly picked from the pool, there is a high chance that they scatter all over the map, thus capturing the speed information of the whole area. This makes the application practical in most city areas. In the next experiment, we demonstrate the estimation of the

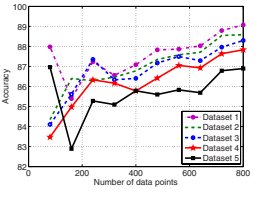
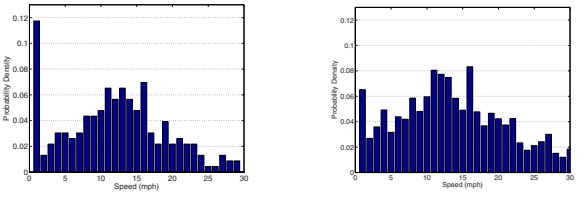


Fig. 6. Accuracy of the density reconstruction

community speed distribution. This community speed distribution can be useful in determining the average speed in the area or compute the percentage of speeding vehicles in that area. To compute the community speed distribution $f(v)$, we marginalize the estimated discrete joint distribution $f(x, y, v)$ as follow

$$f(v) = \sum_{x=1}^{30} \sum_{y=1}^{30} f(x, y, v) \Delta_{XY} \tag{11}$$

where $\Delta_{XY} = (300/30) * (300/30)$ is the area of a two dimensional bin in XY dimension. Figure 7(a) and 7(b) shows the real community speed distribution and the estimated community speed distribution, respectively. We see that the two speed distributions are similar except for the first bin corresponding to zero speed. This can be explained because the density estimation algorithm tends to produce a smooth distribution. Thus, the speed value of the bin is smoothed out. The percentage of speeding vehicles in the community can be computed as the sum of bins with larger than 25 miles/hr speed. In this case the real community percentage of speeding is about 7% while the estimated percentage of speeding is 10% which is a good estimate.



(a) Real community speed distribution

(b) Reconstructed community speed distribution

Fig. 7. Real and reconstructed speed distribution

6 Conclusion

In this paper, we present theoretical foundations for perturbation based mechanisms for ensuring privacy while allowing correct reconstruction of community statistics of interest. Previous data perturbation techniques fail to ensure either privacy or correct reconstruction of community statistics in the case of correlated multidimensional time-series data. The algorithms proposed in this work allow participants to add noise to multiple correlated data streams prior to sharing in a privacy-preserved way while making sure that relevant community statistics are still reconstructible. A participatory sensing application for traffic monitoring is developed which allows participants to "lie" about their actual location and speed, while letting the community estimate useful traffic statistics (e.g., speed map, percentage of speeding vehicle, etc) with high accuracy.

References

1. Burke, J., others: Participatory sensing. In: Proc. of ACM SenSys. (2006)
2. Hull, B., et al.: Cartel: a distributed mobile sensor computing system. In: Proc. of SenSys., pp. 125–138 (2006)
3. Eisenman, S.B., et al.: The bikenet mobile sensing system for cyclist experience mapping. In: Proc. of SenSys., pp. 87–101 (2007)
4. Miluzzo, E., et al.: Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In: Proc. of SenSys., pp. 337–350 (2008)
5. Ganti, R.K., Pham, N., Tsai, Y., Abdelzaher, T.F.: Poolview: Stream privacy for grassroots participatory sensing. In: Proc. of SenSys., pp. 281–294 (2008)
6. Agrawal, R., Srikant, R.: Privacy preserving data mining. In: Proceedings of the ACM SIGMOD, pp. 439–450 (2000)
7. Agrawal, D., Aggarwal, C.C.: On the design and quantification of privacy preserving data mining algorithms. In: Proc. of ACM SIGMOD, pp. 247–255 (2001)

8. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of Royal Statistical Society B39, 1–38 (1977)
9. Wu, J.: On the convergence properties of the em algorithm. The Annals of Statistics 11(1), 103, 95 (1983)
10. Lian, F.L., Murray, R.: Real-time trajectory generation for the cooperative path planning of multi-vehicle systems. In: Proceedings of the 41st IEEE Conference on Decision and Control, vol. 4, pp. 3766–3769 (2002)
11. Saha, A.K., Johnson, D.B.: Modeling mobility for vehicular ad-hoc networks. In: Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks, pp. 91–92. ACM, New York (2004)
12. Karnadi, F., Mo, Z.H., chan Lan, K.: Rapid generation of realistic mobility models for vanet. In: IEE Wireless Communications and Networking Conference, pp. 2506–2511 (2007)
13. Fiore, M., Harri, J., Filali, F., Bonnet, C.: Vehicular mobility simulation for vanets. In: 40th Annual Simulation Symposium, 2007. ANSS 2007, pp. 301–309 (2007)
14. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
15. Kelsey, J., Schneier, B., Wagner, D., Hall, C.: Cryptanalytic attacks on pseudorandom number generators. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 168–188. Springer, Heidelberg (1998)
16. http://www.netlab.tkk.fi/tutkimus/dtn
17. http://www.garmin.com/products/etrexLegend

A Appendix

A.1 Proof of Theorem 1

We begin with the expansion the auxiliary function Q by noting that the data points are i.i.d.

$$\begin{aligned} Q(\Theta|\hat{\Theta}^k) &= E_{X|Y}\left[\log f_{X;\Theta}(\bar{X})|\bar{Y}, \hat{\Theta}^k\right] \\ &= E_{X|Y}\left[\log \prod_{j=1}^{N} f_{X;\Theta}(x_j)|y_j, \hat{\Theta}^k\right] \\ &= \sum_{j=1}^{N} \int_{\Omega} \log f_{X;\Theta}(\gamma) f_{X|Y;\hat{\Theta}^k}(\gamma|y_j) d\gamma \end{aligned}$$

In the last step, the expectation is taken over all possible values of X given the observation y_i. We further expand the auxiliary function Q using Bayes' formula and the fact that $f_{Y|X}(Y|X) = f_N(Y - X)$ because $N = Y - X$.

$$Q(\Theta|\hat{\Theta}^k) = \sum_{j=1}^{N} \int_{\Omega} \log f_{X;\Theta}(\gamma) \frac{f_{XY;\hat{\Theta}^k}(\gamma, y_j)}{f_{Y;\hat{\Theta}^k}(y_j)} d\gamma$$

$$= \sum_{j=1}^{N} \frac{1}{f_{Y;\hat{\Theta}^k}(y_j)} \int_{\Omega} \log f_{X;\Theta}(\gamma) f_{X;\hat{\Theta}^k}(\gamma) f_N(y_j - \gamma) d\gamma$$

$$= \sum_{j=1}^{N} \frac{1}{f_{Y;\hat{\Theta}^k}(y_j)} \sum_{\omega_I} \int_{\omega_I} \log(\theta_{\omega_I}) \hat{\theta}^k_{\omega_I} f_N(y_j - \gamma) d\gamma$$

In the last equation, the integral over the Ω is discretized and is computed as the sum of the integral over all subspaces ω_I in which the value of the discrete density function is constant. Also the value of $f_{Y;\hat{\Theta}^k}(y_j)$ is computed as follow:

$$f_{Y;\hat{\Theta}^k}(y_j) = \int_{\Omega} f_Y(y_j|x) f_{X;\hat{\Theta}^k}(x) dx$$
$$= \sum_{\omega_I} f_N(y_j - \omega_I) \hat{\theta}^k_{\omega_I}$$

$$Q(\Theta|\hat{\Theta}^k) = \sum_{j=1}^{N} \frac{1}{f_{Y;\hat{\Theta}^k}(y_j)} \sum_{\omega_I} \hat{\theta}^k_{\omega_I} \log(\theta_{\omega_I}) \int_{\omega_I} f_N(y_j - \gamma) d\gamma$$

$$= \sum_{\omega_I} \hat{\theta}^k_{\omega_I} \log(\theta_{\omega_I}) \sum_{j=1}^{N} \frac{f_N(y_j - \omega_I)}{f_{Y;\hat{\Theta}^k}(y_j)}$$

$$= \sum_{\omega_I} \hat{\theta}^k_{\omega_I} \log(\theta_{\omega_I}) \phi^k_{\omega_I} \qquad \square$$

A.2 Proof of Theorem 2

This is an optimization problem with a constraint which ensures that Θ is a proper density function.

$$\hat{\Theta}^{k+1} = \operatorname*{argmax}_{\Theta} Q(\Theta|\hat{\Theta}^k)$$
$$\sum_{\omega_I} \theta_{\omega_I} m_{\omega_I} - 1 = 0$$

The Lagrangian of the optimization is given by

$$L(\theta_{\omega_I}, \lambda) = Q(\Theta|\hat{\Theta}^k) + \lambda(\sum_{\omega_I} \theta_{\omega_I} m_{\omega_I} - 1)$$
$$= \sum_{\omega_I} \hat{\theta}^k_{\omega_I} \log(\theta_{\omega_I}) \phi^k_{\omega_I} + \lambda(\sum_{\omega_I} \theta_{\omega_I} m_{\omega_I} - 1)$$

The optimized values $\hat{\theta}_{\omega_I}^{k+1}$ satisfied $\frac{\partial L}{\partial \theta_{\omega_I}}(\hat{\theta}_{\omega_I}^{k+1}) = 0$ and $\frac{\partial L}{\partial \lambda}(\hat{\theta}_{\omega_I}^{k+1}) = 0$. After some algebraic transformation we get

$$\lambda = -\frac{1}{N}\sum_{j=1}^{N}\frac{1}{f_{Y;\hat{\Theta}^k}(y_j)}\sum_{\omega_I}\hat{\theta}_{\omega_I}^{k} f_N(y_j - \omega_I) \tag{12}$$

Since $Y = X + N$ thus the density of Y is the convolution of the density of X and N. It is straight forward to show that

$$f_{Y;\hat{\Theta}^k}(y_j) = \sum_{\omega_I}\hat{\theta}_{\omega_I}^{k} f_N(y_j - \omega_I) \tag{13}$$

Substitute (13) into (12) yield $\lambda = -1$. Therefore

$$\hat{\theta}_{\omega_I}^{k+1} = \frac{\phi_{\omega_I}^{k}}{m_{\omega_I}}\hat{\theta}_{\omega_I}^{k} \qquad \square$$

Gathering Sensor Data in Home Networks with IPFIX

Thomas Kothmayr, Corinna Schmitt, Lothar Braun, and Georg Carle

Institut für Informatik, Technische Universität München
Garching bei München, Germany
kothmayr@in.tum.de, {schmitt,braun,carle}@net.in.tum.de

Abstract. New developments in military, health and home areas call for new approaches for data acquisition in real-time. Such application areas frequently include challenging requirements for collection, processing and analysis of environmental data. Wireless Sensor Networks can collect such environmental data efficiently. Collected sensor node data needs to be transmitted in an efficient way due to limitations of sensor node resources in battery power and available bandwidth. In this paper, we present a method for efficient transmission of sensor measurement data using the IETF standard IPFIX. We show that its template based design is suitable for efficient transmission of senor data with low bandwidth consumption. In this paper, we present the protocol and its implementation in Wireless Sensor Networks (WSNs). Additionally, a header compression scheme is introduced which further reduces communication cost during data transmission.

1 Introduction

Research efforts for wireless sensor technologies become more and more important due to the number of devices in use. Common sensor nodes are only equipped with low-cost hardware and are limited in available bandwidth, memory and battery power. Therefore, communication within a sensor network needs to be very efficient. As bandwidth is limited and data transmission exhausts battery power, transmitting sensor measurement data with little overhead is necessary.

Home networks have an additional requirement. Adding new sensor nodes into a home network should be performed without any (or only minimal) manual reconfiguration of the network. Additionally, Wireless Sensor Networks (WSNs) should be seamlessly integrable into an existing infrastructure.

Concerning resources, similar constraints can be found in the field of network monitoring. Although network monitors are usually equipped with a lot of memory and processing power, they have to observe and process a lot of traffic. Generating, encoding and transmitting information about the observed traffic needs to be implemented at low cost in order to preserve most of the available resources for the monitoring itself.

Therefore, Claise et al. developed the IP Flow Information Export protocol (IPFIX) [3], which is used for transmitting monitoring data. It was standardized

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 131–146, 2010.

by the Internet Engineering Task Force (IETF) in 2008. The protocol was designed to transport flow and packet data, but can also be used for transmitting arbitrary data in an efficient way. It has a template-based concept for encoding measurement data.

In this paper, we present the protocol IPFIX and analyse how it can be used for efficient transmission of sensor data within a Wireless Sensor Network. Furthermore, we will discuss how IPFIX can be embedded into home networks with an infrastructure of wireless sensor nodes.

The remainder of this paper is organized as follows: Section 2 presents the IPFIX protocol and discusses its properties, focusing on the special constrains of wireless sensor nodes. Afterwards, Section 3 describes how IPFIX on wireless sensor nodes can be deployed in the context of IP based home networks. Furthermore, we will discuss how data security methods and compression can be integrated with IPFIX. Afterwards, we will describe our implementation approach of IPFIX in Section 4. Finally, Section 5 will discuss related work before conclusions are drawn in Section 6.

2 The IP Flow Information Export Protocol

2.1 The Protocol

IPFIX [3] was developed by the Internet Engineering Task Force for transmitting flow information between a network monitor and flow data collectors. Communication takes place between an *Exporter* and a *Collector* in IPFIX terminology. IPFIX is specified as a PUSH-Protocol with an exporter periodically transmitting data to one or more collectors.

This design choice seems to be suitable for WSNs because wireless sensor devices tend to disable their wireless network device as long as possible in order to save energy. Aggregating sensor data with a request-response protocols may fail in these scenarios.

For IPFIX, a template based design was developed to exchange measurement data with little overhead. Measurement data is exchanged in so called *Records*. The protocol distinguishes, amongst others, between *Template Records* and *Data Records* as shown in Figure 1.

Data Records contain the measured data while Template Records contain meta information about the information which is transmitted in the Data Records. This meta information covers the type and the length of the measurement data. An Exporter sends a Template Record only once to its Collector to announce the structure of the upcoming data records. The Template Record is stored by the Collector for decoding incoming Data Records. An unique ID, called *Template ID*, is assigned with every Template Record and it is sent to the collector. Further Data Records will reference this ID.

As shown in Figure 1, each Template Record describes the encoding of the transmitted sensor measurement values in a Data Record. A Record may contain several several *fields* where a field corresponds to a measurement type like brightness or humidity. Each field in the Template Record describes the type

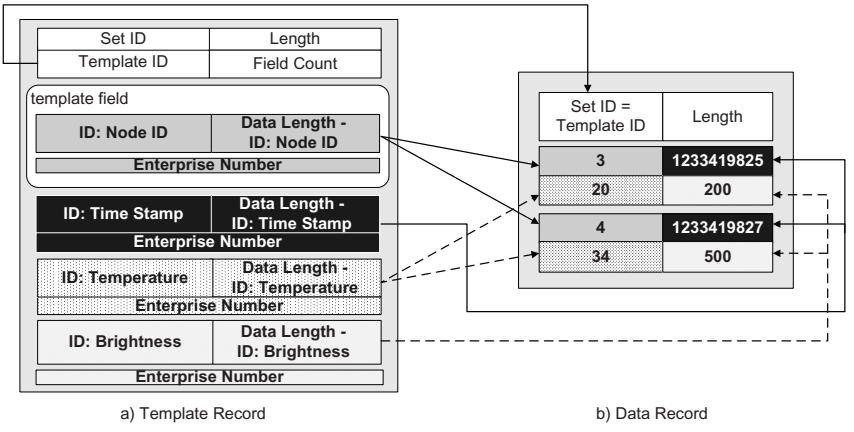


Fig. 1. IPFIX Records with decoding pointers

and length of the corresponding field in the Data Record (Figure 1 shows 4 template fields). The type is uniquely described with a *Type ID* and an *Enterprise ID* in a template field. The Type ID specifies the type of data while the Enterprise ID denotes the organization which issued the Type ID. IPFIX standardized several IDs which are necessary to exchange traffic measurement data like sourceIPv4Address or destinationIPv4Address. The ID field consists of 16 Bits, where IDs 1-32767 are reserved for these traffic measurement data types [21].

If vendors want to exchange different data, for example sensor measurements, new IDs located above ID 32767 must be used. Hence, if the most significant bit for all these IDs is set to 1, a Collector concludes to see a *non standard ID*. In the next step the *Enterprise ID* (EID) will be checked by the Collector in order to find the organization which issued the ID. Each vendor has to register an Enterprise ID with the Internet Assigned Numbers Authority (IANA) [8] which will ensure that any vendor can be uniquely identified. Each vendor can specify up to 32767 own IDs for their data, because 15 bits are left for the Type ID field. We can use this facility to transmit sensor measurement data over IPFIX. It is necessary to register an Enterprise ID for sensor measurement data. Afterwards, we can specify our own standard IDs for common sensor measurement data (e.g. temperature value).

A template field also contains a *length* which announces the length of the transmitted data field in the Data Record. The length is declared in a 16 Bit counter, which allows very long data fields to be included in a Data Record. The fields length is important for a Collector when it decodes a Data Record as Data Records do not include any meta information about the measurement data.

As Figure 1 shows, several template fields may be included into one Template Record, e.g. time stamp and brightness measurement data. If different data should be transmitted to different nodes the Exporter needs more than one

Template Record. Different Template Records are also needed if aggregated and non-aggregated data should be transmitted.

Sensor nodes act as Exporters and transmit their measurement data using IPFIX. When the sensor node boots up, it has to announce a Template in order to announce its measurement data to the Collector. This has to be done only once, as a Collector has to buffer the Template and can use it to decode Data Records. Data Records do not have to contain anything but the measurement data as all meta information has been already sent in the Templates. They only have to contain the number of transported data fields as well as the template ID which is necessary for decoding the record. If a template announces two types of measurement data, e.g. light and temperature, it forms the template record {light, temperature}. Therefore, data records also consist of the tuple {light value, temperature value}. As a consequence, both values need to be included into the record. Several data records can be put into a single message. All records within a packet that can be decoded with a single template, form a so called *Data Set* as shown in Figure 1.

An Exporter transmits a Data Record. The Collector will look up the Template ID and uses the corresponding template to decode the data as illustrated in Figure 1. A Pointer will be hold by the IPFIX parser which points into the Data Set after the length of the Data Set field. The length of the first field will be looked up in the Template Record and the appropriate numbers of bytes will be read. The data type can be identified by its Type ID. Afterwards, the pointer will be advanced by the length of the given field. Then, the next field will be read in the same way.

This template based approach will ensure that meta information about the transmitted data is sent only once. Thus, meta information does not need to be transmitted with every measurement report by the sensor nodes. This in turn results in smaller packets.

Both producing as well as parsing IPFIX Data Records is very easy. The header which contains, amongst others, the Template ID and the number of measurement fields is produced by a sensor node. In the next step the measurement data is packed into the Template in the announced order. A Collector has to read the Template ID and can then read one data field after another as specified in the Template. This process is easy to implement (see Section 4) and has very low processing needs as only pointers need to be moved over the data record. If a pre-defined (hard coded) template is used, this process can be implemented even on very small motes. Multiple templates could be used if the nodes would have more resources, to allow measurement data analysis as done by the base station or servers.

2.2 Identifying Measurement Data of Sensors

Sensor measurement data is identified by the *Type ID* and the *Enterprise ID* (EID). To enhance interoperability they need to be standardized. For today's home networks, typical environmental data can be measured by sensor nodes. Therefore, standard Type IDs can be issued. Up to now, no EID for sensor node

data exists, thus it must be chosen and registered by IANA. This EID can then be used to identify sensor node measurement data. Also, new IDs describing typical sensor data as shown in Table 1 must be standardized. Semantics and type length need to be included in the ID standardization in order to ensure interoperability. New generations of sensor nodes will have the ability to measure other types of data which will result in new IDs. Each vendor can register their own EID and specify their own IDs if he wants to include proprietary data types. However, as the common base for transmitting data is still IPFIX, IPFIX interoperability between devices in the network is enhanced.

Table 1. Possible IDs Space for Sensor Measurement Data

ID	Purpose	Length	Range
1	Node-ID	2 bytes	0 - 65535
2	Temperature	2 bytes	-40 - 123.8C
3	Seismic Data	2 bytes	-2g - 2g
4	Brightness	2 bytes	0 - 10000 Lux
5	Humidity	1 byte	0 - 100% RH
6	Barometric Pressure	1 byte	300 - 1100 mbar

2.3 Data Compression and Aggregation on Top of IPFIX

Due to limited resources on sensor nodes, minimizing data during transmission is desired. Therefore, aggregation can be performed on the IPFIX data in order to reduce the overall amount of data. At first, several measurement results from one or several sensor nodes can be aggregated within a single data packet. Therefore, less packets need to be transmitted which saves energy on the sensor nodes. This kind of data aggregation technique works on arbitrary data without considering measurement context. Additional aggregation techniques can be deployed which consider application context as introduced by Przydatek et al. [20]. Aggregator nodes in the WSN need to be equipped with hardware because they have to store the templates of the child nodes. If a WSN is composed of many uniform nodes which use the same template, all nodes can perform data aggregation.

Another possibility to reduce the transmitted data amount is data compression. The authors in [17] showed that flow and packet measurement data can be compressed with simple methods resulting in smaller packet, which further helps to reduce bandwidth consumption. Thus, it can be reasonable to perform compression on sensor measurement data, too. However, this approach focuses on compressing the actual IPFIX payload. Since typical packet sizes in WSNs are small, the IPFIX header introduces a big source of overhead. Therefore we will introduce an approach to minimize this overhead by compressing the IPFIX header in Section 4.

The wireless part of the network must be connected to a wired infrastructure at some point as described in Section 3. This wired infrastructure is usually based on IP. Hence, using IP within the WSN seems to come natural. Additionally,

IPFIX was standardized to work on IP. By using IP in WSNs, wireless nodes can be addressed by nodes in the wired infrastructure. This enables data transmission from the wired infrastructure into the WSN.

In order to optimize IPv6 for the use in WSNs, 6LoWPAN was developed for wireless sensors and was standardized by the IETF [15]. Harvan et al. implemented an 6lowpan/IPv6 stack on top of 802.15.4 networks [7]. 802.15.4 provides two types of addresses with a length of 16 or 64 bit. Depending on the used hardware, the transmitted payload with 6LoWPAN can be up to 127 bytes for one frame. Larger IPv6 packets need to be fragmented in order to be transmitted within the 6LoWPAN network. As IPv6 has a header size of 40 bytes, too much payload size is occupied by header information. Therefore, a header compression scheme has been standardized resulting in a 2 bytes sized 6LoWPAN header. Similar compression mechanism can be used for the transport headers. An 8 bytes sized UDP can be down sized to four bytes using this compression scheme. The IPv6 compression mechanism is called HC1 and the UDP compression mechanism is called HC_UDP. Without this compression, only 50-66 bytes are left for the data payload, depending on the address types in the 802.15.4 Header. With compression, there is space for 94-110 bytes which nearly doubles the available space for payload [15].

3 Application in Home Networks

Wireless Sensor Networks can perform valuable tasks in home networks. Home networks have additional requirements to WSNs, compared to other applications of sensor networks.

One requirement is the seamless integration into the existing infrastructure. Additionally, users might want to buy devices from different vendors and deploy them into their network. Therefore, devices of different vendors should interoperate and integrate themselves into the existing infrastructure of the Wireless Sensor Network. Handcrafted vendor specific protocols are unlikely to fulfill this requirement. Instead, a common standard for data transmission, like IPFIX, is needed to achieve interoperability.

We will now present why IPFIX is a suitable protocol for the deployment in home networks. Furthermore, we will present how to use application aware data compression techniques to reduce the overall data amount in the network.

Our proof-of-concept implementation is implemented in the context of the Eureka Celtic Project "Autonomic Home Networking" (AutHoNe) [2].

3.1 Application Scenario

The sensor network which is deployed in our home network is supposed to collect environmental data. This data comprises temperature and lighting measurements at the moment and is used to control the lighting and temperature conditions within the house.

Therefore, every room contains several sensors which are linked by a low-power IEEE 802.15.4 wireless mesh. As home networks are usually based on the

IP protocol, sensor nodes should support IP, too [5]. This can be achieved by using 6LoWPAN [7], an IPv6 standard for IEEE 802.15.4 networks. As IPFIX was designed to run on top of IP, it is not necessary to adapt IPFIX to work with other network layer protocols. 6LoWPAN is an adequate solution for sensor nodes as it meets sensor node requirements by defining header compression mechanisms for IPv6 packets transmitted over IEEE 802.15-based networks.

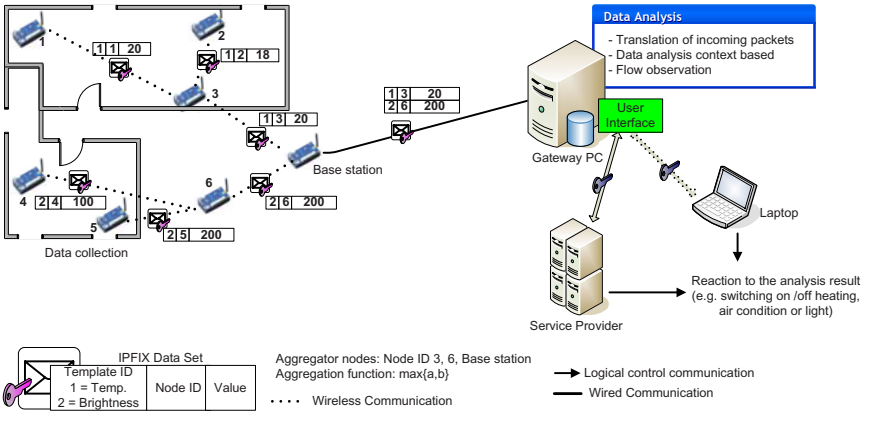


Fig. 2. Overview of application scenario

Figure 2 presents our application scenario. We assume to have several rooms which are equipped with sensor nodes. The sensor nodes are able to measure temperature and light, and are able to build a meshed network to transmit the measurement data to a central server. The server is able to analyse the data and to control the heating and lighting system. For our testbed we use the IRIS motes from Crossbow Technology Inc. [4] as node hardware. The IRIS mote which is used in our setup has the dimensions of 58 x 32 x 7 mm, without the battery pack. Thus, it does not leave much room for the micro controller, flash memory (128kb) and RF transceiver, all of which are located on this board. The available sensor boards have sensors for temperature, brightness and humidity among others.

3.2 IPFIX for Data Transmission

As wireless nodes boot up in the scenario, they will use the 6LoWPAN auto configuration features to obtain an address. Using this address, they will announce their templates to a central server. This server either needs to be configured on the sensor nodes or a special address in the IEEE 802.15.4 can be chosen to address the server.

Afterwards, all nodes in a room measure the current temperature and send their measurement results to the server. During this process, aggregation can be performed by *aggregator nodes*. All nodes that are able to parse IPFIX messages

and have enough resources for holding at least three IPFIX messages in memory can be used as aggregator nodes.

Since transmission is performed on the mesh network, these nodes can aggregate their measurement data with other received measurement data into a single packet. Application specific aggregation for home networks can be performed. In our home network, heat control can be activated for each room depending on the measured temperature. For each room, only minimum, maximum or average temperatures are needed for a decision on whether to turn on the heating or the air condition. IPFIX messages that travel through the network can therefore be aggregated as suggested by Przydatek et al. [20].

Adding devices from different manufactures into the WSN can be done, if they support IPFIX. If all of them use only standard IDs, interoperability between all devices is ensured. If some device vendor wants to specify their own data format, they can register their own EID and issue own IDs. These devices can still be integrated into the home network, as other nodes in the WSN do not need to know the semantics of the new IDs.

3.3 Security in IPFIX Transmissions

Measurement data security and data integrity can be integrated as well. IPFIX copes with these security issues by specifying that every IPFIX device needs to support TLS (on stream based transport protocols) or DTLS (on datagram based transport protocols). Fouladgar et al. developed Tiny 3-TLS [6], a TLS handshake sub-protocol for sensor nodes, which can be used for securing IPFIX data transmission. This conforms to the security considerations from IPFIX.

Other protocols can also be used to assure data security and message authentication in WSNs. TinySec [9], for example, offers an encryption mode where data payload is encrypted and the packet itself is authenticated by a MAC. Another approach using the same idea as TinySec was developed by Luk et al. [13], called MiniSec. It is a secure sensor network communication architecture which modifies the common packet structure of TinyOS and combines features from TinySec and ZigBee [22] to perform low energy consumption and high security.

These protocols can be used instead of TLS, if an existing WSN already implements one of these protocols. However, using TLS is highly recommended.

4 Implementation of IPFIX for Wireless Sensors

In this section we want to characterize the problems and challenges we need to face during the implementation of IPFIX for Wireless Sensors. For Wireless Sensor Networks, two problem fields exist: Environment and Hardware.

In home networks, the environmental problems can be ignored because the network is deployed indoors. We know where each sensor is located and what kind of measurements can be conducted. The distance between the nodes is short, thus no environmental blockage must be taken into account.

Hardware limitations are way more concerning. As described in Section 3 IRIS motes from Crossbow Technology Inc. [4] are used in our application scenario.

These nodes have several limiting factors, such as only 128kb flash memory, 512kb measurement flash and 8kb RAM. Together with the limited power supply of wireless senors the computational capacity is quite limited. These limitations should be kept in mind for the design decisions described in the upcoming section.

4.1 Design Goals and Implementation Decisions

A sensor node has to perform the following tasks:

- Gather data from all sensors.
- Encode measurement results in IPFIX packets and transmit them to the base station.
- Perform in-network aggregation to reduce the amount of network traffic and preserve energy.

The receiving end at the Gateway PC has to perform these tasks: First receiving and parsing IPFIX packets on a Gateway PC must be guaranteed. And secondly the acquired data must be transfered to a home networking infrastructure.

Figure 3 shows all components involved in this process. Both ends, the sensor node as well as the receiving gateway are mapped on the Figure.

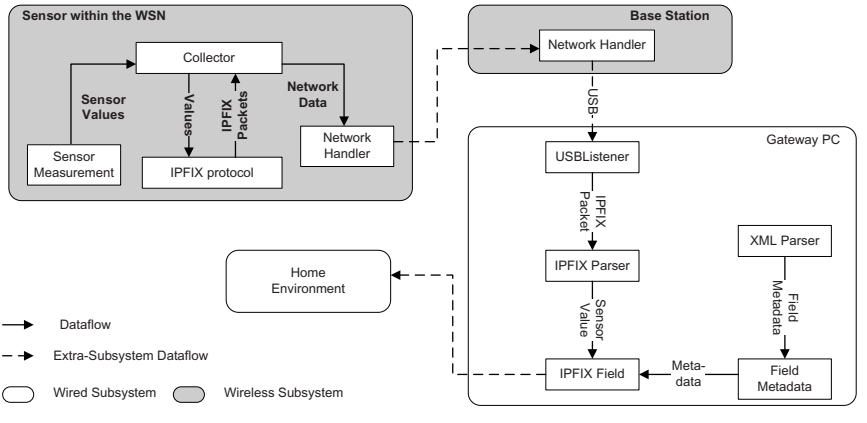


Fig. 3. Data Flow in all components

The node's sensors are queried periodically to generate new sensor data. These raw values are transmitted to the *tinyIPFIX library*, which encodes them into Data Records. The location within the Data Record is specified by an IPFIX template. The template is generated and sent automatically when the node boots. After all sensors have been queried and the IPFIX packet is ready for transmission, it is sent to the *Base Station* via a multihop network. The Base Station listens for incoming packets from nodes in the network and transmits

their payload to the Gateway PC over an USB port. On the Gateway PC, there a Collector waits for transmissions on the USB port. The IPFIX messages sent via USB are parsed according to the matching IPFIX templates, the sensor values are extracted as shown in Figure 1 and transferred to the home environment.

Currently the program for the sensor nodes consists of three main operative components, `ControllerC`, `tinyIPFIXC` and `NetworkHandlerC`. `ControllerC` is the main module of the program, it periodically queries the sensors and passes their reported values to `tinyIPFIXC`, an implementation of IPFIX for TinyOS 2.x. After all connected sensors have reported their values, `ControllerC` receives a byte array containing the finished IPFIX message, which it passes on to `NetworkHandlerC`. `NetworkHandlerC` implements the network communications in a transparent way, so that transmission protocols may be exchanged as needed. Currently, communication is based on the *Collection* protocol of TinyOS. We plan to migrate this to 6LoWPAN in the future. Figure 4 shows a simplified version of the application's wiring. As mentioned in [12], components need to be explicitly wired together.

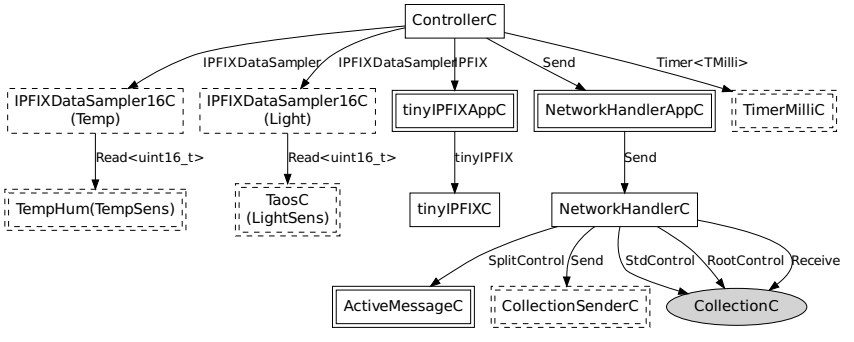


Fig. 4. Simplified wiring of the mote's program

The interface for acquiring sensor data was designed with the following goals: Additional readings need to be added without major changes to the application code, sensor readings should always be linked to an IPFIX *Field* and *Enterprise ID*. Finally, it should be possible to automatically generate IPFIX templates based on the connected sensors.

To generate the IPFIX template, the node queries all connected sensors about their Field ID, Enterprise ID and field length at startup. Similarly, to generate a data record, the node issues a read command to all connected sensors periodically. The sensors return their values after a certain latency and not necessarily in the same order as the read commands were issued. Therefore, a sensor needs to be associated with their respective Field ID, EID and field length.

This was addressed by designing a bidirectional interface called `IPFIXDataSampler` which is implemented by several generic modules. Generic

components can be instantiated with parameters [12]. This allows for a consistent linking of an IPFIX Field ID / Enterprise ID combination with a sensor, since every instance of `IPFIXDataSampler` can report exactly one value. One simply has to pass the according values when creating an instance of the module which provides the interface. `IPFIXDataSampler` defines two commands which are answered by two events. One is `command void report()` with `event void reportBack()` being the according event. `reportBack()` is used to register all providers of `IPFIXDataSampler` with `CollectorC`. Directly after booting, `CollectorC` issues the report command to all connected samplers. When they report back, it uses the information provided by `reportBack()` to create a new field definition in an IPFIX template, thereby addressing the design goal of automatic template creation. The second command is `command read()` which prompts the implementing module to return a reading of the connected sensor. This reading is reported back by `event void readDone()`.

```
 1  configuration ControllerAppC{}
 2  implementation{
 3      components ControllerC as App;
 4      ...
 5      components new IPFIXDataSampler16C(0x80A0,0xF0AA00AA) as Temp;
 6      components new IPFIXDataSampler16C(0x80A2,0xF0AA00AA) as Light;
 7      components new TempHumc() as TempSens, new TaosC() as LightSens;
 8
 9      Temp.Sensor -> TempSens;
10      Light.Sensor -> LightSens;
11
12      App.Sampler -> Temp;
13      App.Sampler -> Light;
14      ...
15  }
16
17  module ControllerC {
18      ...
19      uses interface IPFIXDataSampler as Sampler;
20      ...
21  }
22  implementation {...}
```

Fig. 5. Example of wiring IPFIXDataSampler providers to CollectorC

IPFIX does not transmit the data type of a field, instead it must be recognized based on the respective field ID, so this implementation can ignore the type and simply proceed working with a network order (big endian) byte array. However, functionalities that perform additional computation, such as e.g. mathematical aggregation functions like `SUM()` or `AVG()` must reconstruct the data type. The design goal of flexible extension is addressed by multiple wiring. In traditional languages, the concept of multiple callers to a single method implementation is commonplace. Since nesC interfaces are bidirectional, this also allows for multiple calls to a single method call, meaning multiple methods can be invoked with a single command. The ability to have multiple callers is described as Fan-in and the concept of multiple calls is called Fan-out [12]. By simply wiring multiple components providing `IPFIXDataSampler` to `ControllerC` one can make effective use of the Fan-out concept as shown in Figure 5.

4.2 IPFIX Header Compression

Since IPFIX was designed for conventional networks, some extensions and changes have to be introduced to increase its efficiency in WSNs. Border gateways between the WSN and the wired network need to translated from compressed IPFIX to standard IPFIX. These border gateways are called IPFIX mediators in IPFIX terminology [11].

One of the problems when deploying IPFIX in sensor networks is the overhead introduced by the relatively large header which is at least 20 bytes in size (16 bytes from the Message Header + 4 bytes from the Set header) as is shown in Figure 6. However, the maximum size of a packet being transferred with an IEEE 802.15.4 network is 127 bytes. To address this issue, a header compression scheme was devised, which will be introduced in this section.

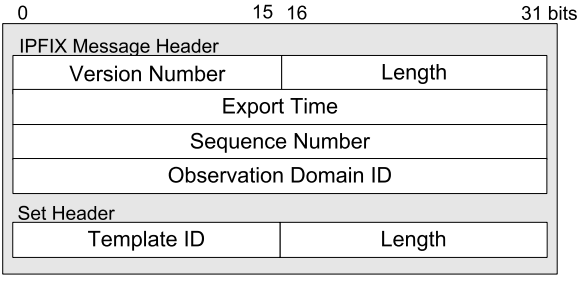


Fig. 6. IPFIX Headers

The idea behind our approach to header compression is to define the length of the fields separately in a pre header which is shown in Figure 7. First the Version field from the original IPFIX header is shortened to 5 bits, this leaves room for the IPFIX version to increase from version 10 to version 31. The definition of the length of the fields *Message Length*, *Export Time*, *Sequence Number* and *Observation Domain ID* follows. A value of 0 in the designated bit(s) means that the field is allowed 1 byte in the subsequent header, a value of 1 means 2 bytes, etc.. The next two bits are designated for the *Template Offset*. Decoders of IPFIX messages are expected to keep track of the sequence in which they received templates from the IPFIX exporters. A value of 0 in the Template Offset bits means that the decoder should use the template it has received last, a value of 1 means the template before that and a value of 2 means two templates before the last one. If this offset is given for a data message, 2 bytes for the Set ID can be saved. If template offset is set to 3 (both bits are one) it is ignored and a proper statement of the template ID is expected in the header. The next bit is called the *Single Set Flag*. It indicates whether the message contains only a single IPFIX set. If this is the case, the explicit statement of set length in the header can be omitted since this value can be computed from the total message length. The last bit in the pre header is the *Template Set Flag*. If it is set to one, the first set in the message is a template set which is defined to have Set ID = 2. Thus, the two bytes for definition of the set ID can be omitted.

In the best case scenario, all header fields can be fitted to 1 byte and the Set Header can be fully omitted. The possibility to shorten the Message Length and Observation Domain ID to 1 byte is fairly obvious. Most messages will be shorter than 255 bytes, in fact if they are transmitted in a single packet, they have to be smaller than 127 bytes with our hardware. Since the Observation Domain ID usually refers to the *Node ID*, a value of 1 byte can accommodate 256 nodes which represents a WSN of medium scale. The Sequence Number can also be shortened to 1 byte, since a rollover after 255 messages is non problematic due to the low data sampling rate of typical WSNs. For the time stamp, a value of 1 byte could refer to the time that has passed since the last full UTC time stamp has been sent. Since the field length can be different with every package sent, it is possible to only transmit a full 4 bytes time stamp periodically and suffice with a delta value in between. For the best case, this method can achieve a reduction in header size from 20 bytes to 6 bytes, or a compression of 81, 25%. Figure 8 gives an example of the best case, which is actually fairly common since it shows the transmission of a Data Record referencing the last sent template set. In the worst case however, header size may increase to 33 bytes when all header fields are defined to be their original length.

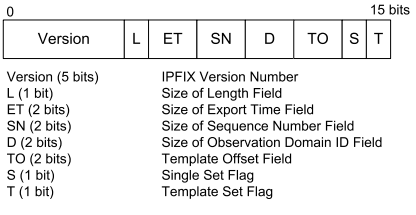


Fig. 7. The IPFIX pre header defining the length of the subsequent header

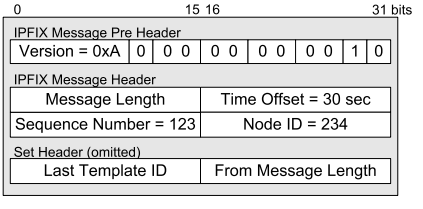


Fig. 8. Best case header for the IPFIX header compression

4.3 Receiving End

To process the data from the WSN to the interface of a home network environment, a gateway is needed. It must parse incoming IPFIX data according to templates generated by the nodes, enrich the received data with meta information (e.g.data type, storage location, etc.) and convert sensor specific values to a general, more abstract data type. In our implementation meta information is stored in a XML-file and fields are matched via Field ID and Enterprise ID.

When an IPFIX template is received, the Gateway creates a new instance of `Field` for every data item defined in the template, based on available meta information. Each instance may contain information about the data type of the field, it's name (for pretty printing), a flag whether or not updates should be passed on to the home network and a simple formula that can be used to perform computations on the received value. Formulas currently support addition, subtraction,

multiplication, division and square roots. They may contain a variable x which is substituted for the received value when the expression is evaluated.

5 Related Work

In 2003 ZigBee was developed for wireless personal area networks [22]. It is a communication protocol based on the IEEE 802.15.4 standard. It was developed for small-scale isolated ad-hoc networks and limited to a single radio standard. Today it is a standard which is used nearly everywhere. But it requires more resources than the 6LoWPAN approach we are using as described in 2.3. ZigBee has a code size with mesh of 32-64K, requires 8K RAM, produces 8-16 bytes overhead, and supports 802.15.4 and no transport layer. 6LoWPAN has a code size with mesh of 22K, requires only 4K RAM, produces only 2-11 bytes overhead, and supports 802.15.4++ and UDP/TCP [16]. Finally, 6LoWPAN requires less resources than ZigBee, thus more resources are left for additional computations and transmissions.

In contrast, 6LoWPAN was developed for scalable networks as an end-to-end part of the Internet. It is applicable to any low-power and low-rate wireless radio. The used IP protocols tie together heterogeneous networks. ZigBee itself is not a standard, it is a special interest group, called ZigBee Alliance [22]. The IETF supports open, long-lived standards and this will be archived by 6LoWPAN which works with modified IPv6 protocols and stacks. Together with the home network scenario using IP addresses for communication we decided to implement 6LoWPAN on the IRIS motes.

As Kimura and Latifi discussed in [10], many algorithms for data compression exist but cannot adapt to the constraints of Wireless Sensor Networks. Thus, special algorithms were developed to compress the transmitted data like Coding by Ordering, Pipelined In-Network Compression, Low-complexity Video Compression and Distributed Compression.

The basic idea of the algorithm *Coding by Ordering* [18] is to drop data at the aggregation node. This can happen if the transmitted data is unique, and the order is irrelevant for the application. Now it is possible to use the transmitting order to transmit additional information to the receiver. This algorithm can be provided by an aggregator node in the network.

Arici et al. developed an compression algorithm called *Pipelined In-Network Compression* in 2003 [1]. This algorithm is also based on aggregator functionality. The sensor measurements are sent to an aggregator node and buffered. During the buffer period the incoming packets are combined and redundant data is deleted before ongoing transmission. The transmitted data uses a shared prefix which can be used for node IDs and time stamps to reduce space in new packets. Depending on the prefix length the data compression can be quite efficient.

The algorithms *Low-Complexity Video Compression* [14] and *Distributed Compression* [19] deal with data compression of visual data. the first algorithm is based on block changing and JPEG data compression. The second algorithm deals with the usage of side information to encode source information. This compression scheme can be applied to lossless and lossy compression schemes.

6 Conclusion

In this paper we introduced a concept to connect a wireless infrastructure to a wired home network scenario. This can be achieved by implementing 6LoWPAN on the sensor nodes to bring IP communication to a wireless infrastructure. In the next step we integrated IPFIX into the WSN and showed the applicableness for home networks in cooperation with 6LoWPAN.

At first, IPFIX defines a efficient data format for transmitting sensor measurement data using low bandwidth. Generating and parsing IPFIX data can be performed with little processing power, thus saving energy on the nodes. Arbitrary aggregation techniques can be deployed to further reduce the transmitted data.

If standard template IDs are issued, interoperability between different devices from different manufacturers can be ensured. At the same time, vendors can register its own enterprise and type IDs to build custom devices. These devices can still interoperate with other devices. By using IP on the network layer below IPFIX, wireless sensor networks can easily be integrated in existing home networks. Therefore, new sensor nodes can be easily deployed and new functionality to the network can be added in an automatic fashion.

To reduce the amount of data traffic within the network and to reduce the energy consumption of the network we introduced a concept of header compression for IPFIX and combined it with header compression of 6LoWPAN to increase the payload capability of each packet. Those compression functions can be combined with aggregation algorithms to gain more efficiency in the transmissions.

Acknowledgment

The presented work is part of the AutHoNe project which is partly funded by the German Federal Ministry of Education and Research under grant agreement no. 01BN070[2-5]. The project is being carried out as part of the CELTIC initiative within the EUREKA framework.

References

1. Arici, T., Gedik, B., Altunbasak, Y., Liu, L.: PINCO: a pipelined in-network compression scheme for data collection in wireless sensor networks. In: Proceedings of the 12th International Conference on Computer Communications and Networks (ICCCN), October 2003, pp. 539–544 (2003)
2. Autonomic Home Networking DE Project Page (2009), http://www.authone.de
3. Claise, B., Bryant, S., Sadasivan, G., Leinen, S., Dietz, T., Trammell, B.H.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information (RFC 5101). Technical report, The Internet Engineering Task Force (IETF) (January 2008)
4. Crossbow Technologies Inc. (2009), http://www.xbow.com/
5. Das, K.: IPv6 and Wireless Sensor Networks. IPv6.com Tech. Spotlight (2008)

6. Fouladgar, S., Mainaud, B., Masmoudi, K., Afifi, H.: Tiny 3-TLS: A trust delegation protocol for wireless sensor networks. In: Buttyán, L., Gligor, V.D., Westhoff, D. (eds.) ESAS 2006. LNCS, vol. 4357, pp. 32–42. Springer, Heidelberg (2006)
7. Harvan, M., Schönwälder, J.: TinyOS Motes on the Internet: IPv6 over 802.15.4 (6lowpan). PIK - Praxis der Informationsverarbeitung und Kommunikation 31(4), 244–251 (2008)
8. Internet Assigned Numbers Authority (2009), http://www.iana.org/
9. Karlof, C., Sastry, N., Wagner, D.: TinySec: a link layer security architecture for wireless sensor networks. In: Proceedings of the 2nd international conference on Embedded networked sensor systems, pp. 162–175. ACM, New York (2004)
10. Kimura, N., Latifi, S.: A survey on data compression in wireless sensor networks. In: Proceeding of the International Conference on Information Technology: Coding and Computing (ITCC), April 2005, vol. 2, pp. 8–13 (2005)
11. Kobayashi, A., Blaise, B., Ishibashi, K.: IPFIX Mediation: Framework. Technical report, The Internet Engineering Task Force (IETF) (October 2009)
12. Levis, P., Gay, D.: TinyOS Programming (July 2009)
13. Luk, M., Mezzour, G., Perrig, A., Gligor, V.: MiniSec: a secure sensor network communication architecture. In: IPSN 2007: Proceedings of the 6th international conference on Information processing in sensor networks, pp. 479–488. ACM, New York (2007)
14. Magli, E., Mancin, M., Merello, L.: Low-complexity video compression for wireless sensor networks. In: Proceedings of the International Conference on Multimedia and Expo (ICME), Washington, DC, USA, vol. 3, pp. 585–588. IEEE Computer Society, Los Alamitos (2003)
15. Montenegro, G., Kushalnagar, N., Hui, J., Culler, D.: IPv6 over Low Power Wireless Personal Area Networks (6LowPAN) - RFC 4944. Technical report, The Internet Engineering Task Force (IETF) (September 2007)
16. Mulligan, G.: The 6lowpan architecture. In: Proceedings of the 4th workshop on Embedded networked sensors (EmNets), pp. 78–82. ACM, New York (2007)
17. Münz, G., Braun, L.: Lossless Compression for IP Flow Information Export (IPFIX). The Internet Engineering Task Force (IETF), Internet-Draft (work in progress), draft-muenz-ipfix-compression-00 (2008)
18. Petrovic, D., Shah, R.C., Ramchandran, K., Rabaey, J.: Data funneling: routing with aggregation and compression for wireless sensor networks. In: Proceedings of 1st IEEE International Workshop on Sensor Network Protocols and Applications, May 2003, pp. 156–162 (2003)
19. Pradhan, S.S., Kusuma, J., Ramchandran, K.: Distributed Compression In Dense Sensor Networks. IEEE Signal Processing Magazine 19, 51–60 (2002)
20. Przydatek, B., Song, D., Perrig, A.: SIA: Secure information aggregation in sensor networks. J. Comput. Secur. 15(1), 69–102 (2007)
21. Quittek, J., Bryant, S., Claise, B., Aitken, B., Meyer, J.: Information Model for IP Flow Information Export, RFC 5102 (2008)
22. ZigBee Alliance. ZigBee specification. Technical Report. Document 053474r06 Version 1.0, ZigBee Alliance (June 2005)

Sensing for Stride Information of Sprinters

Lawrence Cheng[1], Huiling Tan[2], Gregor Kuntze[3], Kyle Roskilly[2], John Lowe[2], Ian N. Bezodis[3], Stephen Hailes[1], Alan Wilson[2], and David G. Kerwin[3]

[1] University College London, Computer Science Department, Malet Place, London, WC1E 6BT, UK
{l.cheng,s.hailes}@cs.ucl.ac.uk
[2] Royal Veterinary College, Structure and Motion Lab, Hawkshead Lane, Herts, AL9 7TA, UK
{htan,kroskilly,jlowe,awilson}@rvc.ac.uk
[3] University of Wales Institute, Cardiff, Cardiff School of Sport, Cyncoed Road, Cardiff, CF23 6XD, UK
{gkuntze,ibezodis,dkerwin}@uwic.ac.uk

Abstract. Accurate sprint-related information, such as stride times, stance times, stride lengths, continuous Centre-of-Mass (CoM) displacements and split times of sprinters are important to both sprint coaches and biomechanics researchers. These information are traditionally captured using camera-based systems which are very expensive and time-consuming to setup. This paper investigates - through a series of experiments - whether an integrated sensing system would provide a practical, cost-effective alternative to measuring stride-related information of sprinters. The results show that the system achieves an accuracy within 5ms for stance time and stride time measurements, and ~10cm for localisation-related information such as CoM forward displacement and CoM stride displacement (i.e. stride length).

Keywords: Application, performance monitoring, sports, stride information, wireless sensing.

1 Introduction

Sprint performance is ultimately evaluated by one factor: speed. There are several well known factors that affect sprinters' speed, such as stride length, stride frequency, touch-down and take-off angles[1], etc. [10]. Existing biomechanics research on stride analysis use camera-based systems to capture stride-related information. Although these systems are highly accurate however, they are expensive and time consuming to setup. Pervasive

[1] A stride is the action between subsequent foot-on events of the same foot of a sprinter during a sprint. Stride length is the corresponding distance between each foot-on. Stride frequency is the rate at which a stride is made. Touch-down and take-off angles refer to the angle of the toe in relation to the Centre of Mass (CoM) of a sprinter at foot-on and foot-off respectively. Stride time is the time in-between each foot-on; stance time is the time when the foot is on the ground.

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 147–161, 2010.

computing for sprint (or sports) performance monitoring [1][2][3][8][9] is a relatively new research area. SEnsing for Sports And Managed Exercise (SESAME) [1][2][3] is a multi-disciplinary project to *develop practical, deployable, and inexpensive wired and wireless sensor-network-based systems to support sprint coaching and biomechanics research*. The SESAME Integrated System (IS) includes several wired and wireless track-side and on-body sensing sub-systems: a radio-based continuous speed/location tracking system, a Light Gate (LG)-based split time measurement system, and a wireless foot pressure sensing system. The SESAMS IS was developed to investigate whether sensor technologies are suitable and capable of delivering stride-related information, such as stance time, stride time, stride length, etc. to support coaching and biomechanics research. In this paper, the design and experiment results of the SESAME IS are presented and discussed.

This paper is organised as follows: first, related work and system requirements are discussed; secondly, an overview of the SESAME IS and its sub-systems, and the integration and synchronisation methods are presented; thirdly, the experimental procedure is presented; fourthly, the experiment results are analysed and discussed. Then, the applicability and impact of the system is discussed. The paper ends with a conclusion and future work.

2 Background

Existing biomechanics research studies have been using motion-capture camera-based systems, such as active marker systems (e.g. CODA [6]), or passive marker systems (e.g. Vicon [11] or Qualisys [12]), or high-speed video cameras to obtain stride-related information of sprinters. Although these systems are considered as gold-standard technologies due to their high level of accuracy, and that they provide 3D coverage of (all) body segments, they are very expensive, have limited viewing angle, and are time consuming to setup. It is therefore important to investigate other approaches. An overview of ubiquitous computing for sports performance monitoring was presented in [8]; the work focused on outlining the best practice for designing and implementing ubiquitous computing systems for sports performance monitoring. In [13], a wearable piezoelectric force sensing system for detecting scoring kicks in Taewondo matches was presented. A light-sensor-network-based split time[2] measuring system for sprinting was presented in [3]; the system covers five lanes over a 60m indoor track, and reports (and records) the split times of multiple competing athletes to coaches in real-time. Coaches could then use the split time information to adjust his/her training methods during a training day-session and/or over a training season.

It is argued that, given the limitation in sensor technologies' accuracy and space limitation for on-body sensor attachment, accurate stride-related information could only be derived from fusing together multiple sources of data from both track-side and on-body sensing systems. Thus, this paper focuses on investigating whether an integrated sensing system would provide a practical solution to delivering stride-related information of sprinters.

[2] A 10m split time is the time it takes for a sprinter to sprint 10m.

3 The SESAME Integrated System (IS)

3.1 System Description

Each of the sub-systems must provide unique information that can be fused together to provide meaningful results. More specifically, the following type of information would be needed in order to derive a complete set of stride-related information for supporting coaching and biomechanics research work (in addition to gold-standard data for evaluation): continuous speed (or 1D location) information of the sprinter, stance time, stride time, and additional data for correcting noisy localisation data. It should be noted that, in order to fuse the above data together, all sub-systems must be synchronised. Integration and synchronisation in the SESAME IS is discussed in more detail in section 3.2. All the experiments were conducted in the five-lane 60m indoor sprint track at the National Indoor Athletics Centre (NIAC), Cardiff, UK. CODA is used as a validation tool for the experiments presented in this paper due to their well-recognised high level of accuracy [7]. CODA can be synchronised with external systems either through TRIG IN or SYNC IN[3].

The SESAME Pisa Light Gate (PLG) system was developed by the SESAME team. It is a novel and cost-effective split time measuring system [3] which is capable of providing in real-time gold-standard comparable split time results of multiple competing athletes to coaches and athletes for coaching support. It is permanently installed at NIAC and has been operational since May 2009. Essentially, 30 retro-reflective LGs (retro-reflective light sensor (RL39-55/30/35/40a/116/126a) from Pepperl + Fuchs) were permanently installed in the roof of the stadium. The LGs point at 57.5° (±0.5°) to the reflective tapes which are placed on the white lanes that separate the five lanes of the indoor 60m track. As an athlete cuts through each light beam, the signal generated at the corresponding LG is timestamped by a sensor node (i.e. a gumstix [16], which is a mini Linux computer that is permanently installed at the track-side), thus the corresponding split times are calculated. All software were custom-written: computational software were written in C and the web-based user interface was written in PHP. The PLG system supports TRIG OUT for external synchronisation; the signal is delivered to other systems through a BNC socket on the system's enclosure. Readers should note that split times have been, traditionally, the fundamental block of sprint performance evaluation [14][15]. The PLG system is therefore specifically designed to provide the type of information that coaches are familiar with and could easily relate to.

Fine-grain type of information such as continuous speed (or 1D location) information of a sprinter during a sprint, stride length, etc. are useful to biomechanics research. To derive this information, accurate measurements of the continuous location of an athlete during a sprint are crucial. Radio-based localisation systems are more cost-effective than laser range finders and have a higher level of automation (that they do not require manual adjustments). A radio-based localisation system is used in the

[3] TRIG IN means the system is capable of timestamping a common voltage input trigger signal using its own clock. SYNC IN means the system is capable of being driven to sample based on a series of incoming SYNC pulses. Most camera-based gold-standard technologies, such as CODA and Qualisys, support these form of synchronisation methods (as well as TRIG OUT and SYNC OUT) for synchronising with external systems.

SESAME IS for continuous speed/location tracking of athletes during their sprints. The SESAME localisation system was built on the nanoLoc (NNL) system from nanotron [5], which operates in the 2.45GHz ISM band. The system uses Time of Arrival (ToA) information from packets exchanged using the Double-Sided Two-Way Ranging (SDS-TWR) protocol, between a track-side anchor and an on-body tag to estimate the distance between the two devices. The Peer-to-Peer (P2P) system (which requires just one anchor) was used for the experiments presented in this paper due to: a) its simplicity: only one anchor would be needed at the end of the track and the tag is attached to the subject's CoM (i.e. lower back). Both devices are small in size, thus the disturbance caused by the presence of track-side equipment to other track users is kept to a minimal comparing to a multiple-anchor system; and b) the interest of this paper lies within 1D localisation, which is provided by the P2P system. The sampling rate of NNL was ~100Hz. The timestamping mechanism of the NNL system was modified in order to support TRIG IN: the NNL anchor is connected to a track-side laptop, the latter is also connected to the PLG system via BNC cables. The TRIG OUT signal from the PLG system and the calculated distance results from the NNL system is timestamped at the laptop which provides a common time base for the common trigger signal from the PLG system and the NNL samples. The localisation software were written in C.

The wireless SESAME Force Sensing Resistor (FSR) system is a custom-built foot contact time measurement system. It was designed around Interlink Electronics FSR model 406. The sensors used were thin 1.5" square sensors attached at the heel, mid-foot and toe positions on a standard shoe insole. The general purpose logging board incorporates a wireless transceiver, which gives the capability to synchronise data from multiple boards in disparate locations both track-side and on-body. This is achieved with the addition of a beacon transmitter board to the system, which is located at the track side and has an effective transmission range over 60m. The beacon transmitter sends an incrementing single byte value at a rate of 1Hz. This 'beacon' is received by all logger boards within transmission range and recorded alongside the next ADC sample, which is timestamped by the internal clock. The difference in latency in receiving and processing the beacon among different boards is anticipated to be sufficiently small, so that it is possible to synchronise ADC data from multiple boards with at most one sample interval of error. This is also assuming that the internal clocks do not drift significantly over a 1 second period and that logging has been started on all boards within the rollover period of the beacon value, which are reasonable assumptions. The built-in synchronisation features of the logging boards provide a simple method to combine data from on-body equipment, like the FSR system, with other track-side equipment. The track-side beacon transmitter and 'sync' logger are used for this purpose. The 'sync' logger can record trigger signals from any track-side equipment, such as the TRIG OUT signal from the PLG system, and therefore provides a route to synchronisation with on-body equipment.

3.2 System Integration and Synchronisation

All sub-systems must be synchronised. It should be noted that one unique feature of sprinting experiments is that the experiment runtime is very short, and there is no need to capture stride information beyond each sprint. Thus, a *generic* synchronisation method should be adopted in the SESAME IS to provide easy integration with

future sub-systems. Since crystal clocks drift linearly and the experiment runtime is very short-span, the effect of clock drift is minimal. TRIG OUT (from PLG) is therefore chosen as the cross-subsystem synchronisation method in the SESAME IS. The common trigger is delivered to all track-side sub-systems through BNC cables. Under such arrangement, flexibility to develop individual sub-system is enhanced. Fig. 1 shows the SESAME IS.

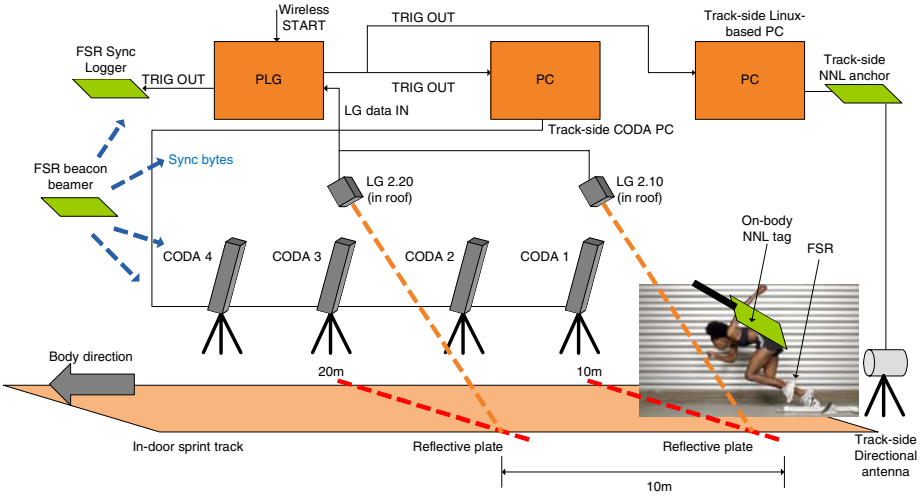


Fig. 1. The SESAME Integrated System

4 Experiment Setup

The purpose of the experiment is to collect data from the SESAME IS across all phases of sprinting, namely: the acceleration phase, the secondary acceleration phase, the maximum speed phase, and the fatigue phase. Due to the viewing angle limitation of CODA, the experiments were divided into four sets:

- Set 0: FSR pressure-sensing and synchronisation accuracy against Force Plates (FPs); CODA was not used because it does not enable one to work out the precise time moments when a specific part of a foot is on the ground
- Set 1: ~0m to ~15m (i.e. acceleration and secondary acceleration phase); including FSR, NNL, and CODA
- Set 2: ~20m to ~35m (i.e. secondary acceleration and/or maximum speed phase); including FSR, NNL, and CODA
- Set 3: ~40m to ~55m (i.e. maximum speed and/or fatigue phase); including FSR, NNL, and CODA

In set 0, the foot-on and foot-off time are detected based on data from the insole-mounted FSR; then, the accuracy of the foot strike timings were compared with those measured using force plates (Kistler Instrumente AG, Winterthur, Switzerland),

which is a gold standard for measuring ground reaction forces. In set 1 to 3, the four CODA scanners were placed on the track-side to monitor full 2D body movement of each subject (i.e. the vertical and forward plane) over the specified area. The sampling rate of CODA was 400Hz. Ten active markers were attached to each subject: right toe, right foot, right ankle, right knee, right hip, right shoulder, right elbow, right wrist, left toe and left foot. These markers would enable CODA to reconstruct the athlete's motion in full (i.e. 2D). The NNL tag is attached to the CoM of each subject (i.e. lower back). The height of the tag relative to ground was measured; the NNL anchor was placed 2.8m *behind* the 0m line (which is the furthest the anchor could be placed away from the 0m line), with a 12dBi directional panel antenna placed on a tripod at the same height as the NNL tag. A directional antenna was used to ensure long range coverage (i.e. >100m indoor). The FSR insole is placed underneath the right root of each subject. Two subjects did a total of 18 sprints over a two-day experiment at NIAC. Only one sprinter ran during each trial to ensure most markers were within direct line-of-sight with the CODA scanners; the same reason for using one FSR insole on the right foot for each subject. Note that the hip marker was used as the CoM of the subject.

5 Results and Analysis

In this section, first, the validation results of FSR on stance time and stride time measurements against Force Plates, and its internal sychronisation validation are presented. Secondly, the relationship between CoM stride displacement and stride length is analysed and discussed. Thirdly, a filtering and correction algorithm for correcting noisy and biased localisation data is presented. Then, the experiment results on CoM displacement during a sprint (i.e. CoM stride displacement) and stride length measurements against CODA are presented and discussed.

5.1 FSR Validation Results and Analysis

5.1.1 Stride Time and Stance Time Results and Analysis

The sharp increase in each channel of the raw data from the FSR system indicates the 'touch-down' of the point where the sensor was attached, and the sharp decrease in the raw data indicates the 'take-off' of the point. These sharp increases and decreases were detected by finding the local maximum and minimum of the first-order differentiation of the raw data. For each stride, the first touch-down time among the three sensors should be the foot-on time for the stride, and the last take-off time within the three sensors should be the foot-off time. In running or sprinting, the touch-down and take-off from the toe sensor were used for foot-on and foot-off timings. The stance duration, which is the time the foot is in contact with the ground, was calculated by taking the difference between the foot-off time and foot-on time.

For validation of the FSR, synchronised FSR and force plate data was collected from ninety-five strides at different speeds (3m/s ~6.5m/s). The force place used was 8m in length, on average two strides were collected per trial. The synchronisation method is the same as the one deployed in the SESAME IS, i.e. a TRIG OUT from

Force Plate is delivered to the FSR's Sync Logger via a BNC cable (section 3.2). The standard deviation of the foot-on time and stance durations calculated from FSR is 3.1ms and 4.2ms respectively compared to those from FP, with 50% of the strides having error within 3ms and 80% having the error within 5ms in both foot-on time and stance durations.

5.1.2 FSR Synchronisation Validation Results

The accuracy of the beacon-based synchronisation between the FSR logging board and the Sync logging board was validated in this section. The two logger boards and beacon transmitter were switched on and logging started at a rate of 1000Hz. A simulated trigger signal was generated, consisting of rising and falling edges at arbitrary intervals, and logged directly into a channel on both logger boards simultaneously. The timings of the trigger signal edges from each board were then converted to a common timebase, using the received beacons, and differenced. From 150 trigger edges, the mean absolute timing error was 0.647ms with a standard deviation of 0.715ms. 67.3% of the edges times were no more than 1ms (1 sample period) different, 88% were no more than 1.01ms different, and 100% were no more than 2ms (2 sample periods) different. This is an acceptable result, particularly as it is anticipated that synchronisation accuracy should improve at the reduced sampling rates (300Hz) used by the FSR system.

5.2 Stride Length Results and Analysis

In this section, first, it was investigated – using CODA data – whether there is a relationship between CoM stride displacement and stride length. Then, the accuracy of NNL data was evaluated by comparing it with the CoM forward displacement from CODA's hip marker data, and investigate how NNL's accuracy could be improved by fusing NNL data with other type of SESAME IS data. Assuming a relationship between stride length and CoM stride displacement exists, and that the accuracy of the NNL data can be improved, one could deduce stride length by combining the NNL CoM forward displacement with the foot-on times measured from the FSR data.

5.2.1 Relationship between CoM Stride Displacement and Stride Length

The purpose of this analysis is to investigate whether a relationship exists between the CoM forward displacement during a stride (i.e. CoM stride displacement) and stride length using CODA data. Since sprinters sprint on their toes, the *vertical* displacement of the right toe, *vertical acceleration* of the right toe, and *forward displacement* of the hip marker were used for analysis (see Fig. 2, note that vertical displacement of toe has been scaled down for display purposes). The vertical displacement and vertical acceleration of toe enable one to work out the foot-on times; this is because at foot-on, the vertical displacement of toe is approximately zero (i.e. on the ground); there is also a sharp change in acceleration. The corresponding turning point in acceleration would be the foot-on time. Using these foot-on times, one could work out the corresponding forward displacement of the hip during a stride (i.e. CoM stride displacement). Then, the data was compared with the stride length data from CODA. Foot-on is defined as the first moment when the toe touches the ground.

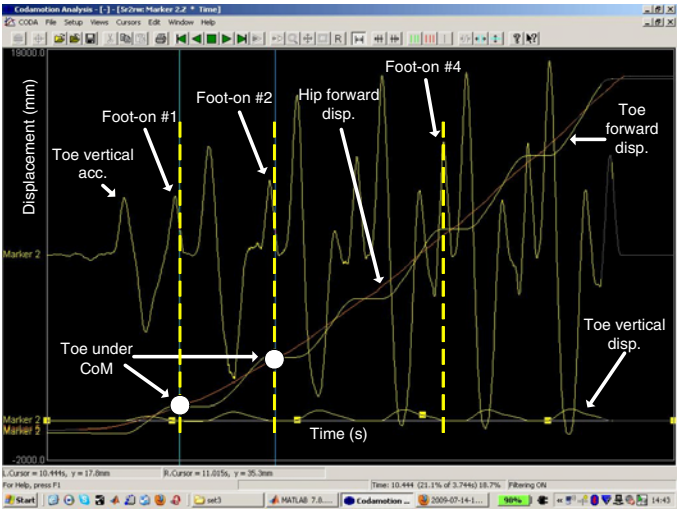


Fig. 2. Toe and hip forward and vertical displacement and acceleration

The results over 18 trials show that for set 1, the difference was 4.491 ± 4.426cm (mean ± SD); for set 2 was 0.683 ± 1.807cm; for set 3 was 0.208 ± 2.295cm. The results suggest that: a) a relationship does exist between CoM stride displacement and stride length, in fact there is a 100% relationship between the two whilst the toe is still on the ground and the CoM moves above the toe (see the first two yellow dotted lines in Fig. 2 from left to right which indicate the times when the hip and the toe are in a straight, vertical line); and b) the two are more closely related as speed increases, this is indicated by the fact that the errors of set 2 and set 3 are significantly less than the errors of set 1. It is concluded that, by combining the foot-on times from FSR with NNL data, the CoM stride displacement can be determined.

5.2.2 Low-Pass Filtering and Bias Correction on Raw NNL Data

In order to derive CoM stride displacement, accurate CoM forward displacement is essential. It is well-known that data from radio-based localisation systems subject to noise and non-constant bias that the bias changes according to distance between the anchor and the tag. Furthermore, one would also anticipate the errors to differ should the surrounding environment changes (e.g. body obstructions, the presence of interfering wireless devices, etc.). For example, after comparing with CODA data, the bias in the raw NNL data was ~6m between ~0m-20m and was ~7m between ~30-40m (Fig. 3 and Fig. 4). These errors are caused by several factors, namely: multiple-path signal reflection from the surrounding environment, background noise, and body obstruction, which are difficult to avoid. One approach is to correct the bias through modeling. However, given the level of variability involved, this approach would be difficult [4].

An option to remove noise in the NNL data would be the use of Kalman filter. However, the assumption of constant speed or constant acceleration cannot be justified because there is no other redundant information on speed or acceleration. The

solution presented in this paper is to remove high frequency noise in the data by low-pass filtering, and to fuse accurate position data from the PLG system (which is available every 10m) with the noisy and non-constant biased NNL location data in order to "correct" NNL's errors. Thus, Fast Fourier Transform (FFT) was used on all the raw NNL data to determine the suitable cut-off frequency for low-pass filtering. The analysis shows that 1Hz is the optimal cut-off frequency.

The PLG system provides accurate location of the subject at set known positions (e.g. 0m, 10m, 20m, etc.). The PLG data was used to correct the filtered NNL data in a piecewise linear model. More specifically, at specific times during a sprint (i.e. when the subject passes each LG), the corresponding NNL measurements are noted[4]. The difference of the two would be the bias and would be used as the correction for all subsequent filtered NNL measurements until the next LG is reached. The corrected NNL data are then compared against the corresponding CODA data.

5.2.3 Error Analysis on Raw, Filtered, and Corrected NNL Data vs. CODA

Fig. 3 to Fig. 5 shows a selection of graphs from the experiment results. The top left-hand subplot of each graph shows how raw NNL data, filtered (1Hz) NNL data, filtered (1Hz) and corrected NNL data (known as corrected NNL data for the rest of the paper), and CODA hip forward displacement measurements (i.e. gold-standard data) change against time. The subplot at top right-hand corner of each graph shows the error distributions of the differences between raw NNL forward displacement data and the corresponding CODA data, the subplot at bottom left-hand corner shows the error distributions of the filtered NNL data (1Hz) against CODA, and the subplot at bottom right-hand corner shows the error distributions of the corrected NNL data against CODA. Results from all sets were presented.

The average mean error of all trials is 5.48cm, with a standard deviation (STD) of 10.82cm. Note that the CODA data uses the hip marker as the CoM; whereas the NNL data refers to the NNL tag which is at lower back, the difference between the two positions was between 5cm to 8cm (i.e. approximately halve the width of the subject's waist); this difference is represented in the mean error. In other words, the error is negligible. More specifically, the STD for set 1, set 2 and set 3 was 12.24cm, 9.68cm, and 9.1cm respectively. It was suggested that the slightly larger STD for set 1 was caused by the bump which occurs recursively at the same location of ~1-2m from 0m, which is ~4-5m from the NNL anchor (Fig. 3).

To identify the cause of the bump, the experiment was repeated after relocating the same set of equipment to the other end of the track (i.e. a different environment); furthermore, a set of static experiments, which involved the subject carrying the tag standing still at various locations for 120s were conducted. The results show that the bump still exists at the same location in relation to the anchor's position (i.e. a repeatable pattern). Another set of identical experiments were carried out but with the directional antenna of the NNL anchor replaced by an omni-directional antenna. It was observed that the bump continue to exist repeatedly but at a different location on the

[4] NNL and PLG are synchronised through a common trigger, but their samples are timestamped using their local clock. Thus, to obtain the corresponding filtered NNL measurements at a specific time, NNL measurements are interpolated. Since NNL has a high sampling frequency (~100Hz), the error of interpolation is minimal.

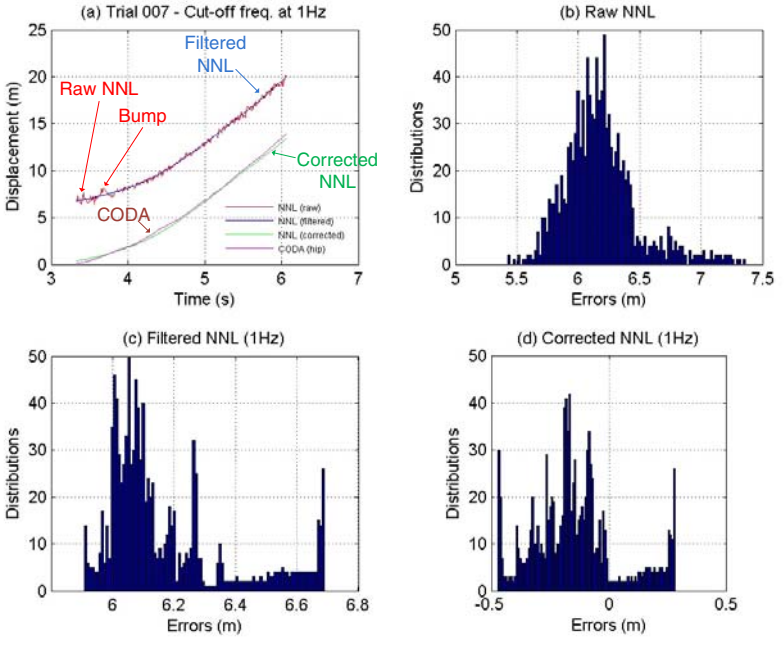


Fig. 3. Error distributions of raw NNL, filtered NNL (1Hz), and corrected NNL vs. CODA (trial 7, set 1)

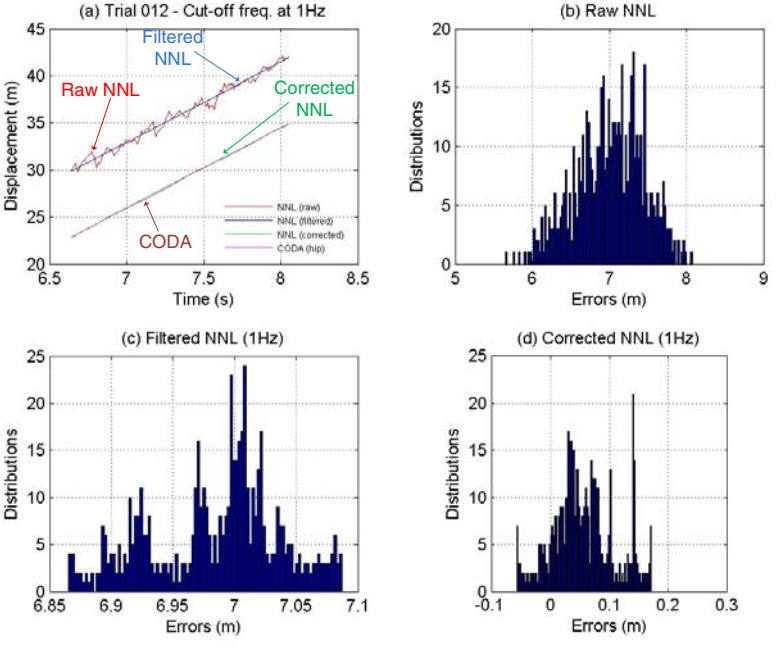


Fig. 4. Error distributions of raw NNL, filtered NNL (1Hz), and corrected NNL vs. CODA (trial 12, set 2)

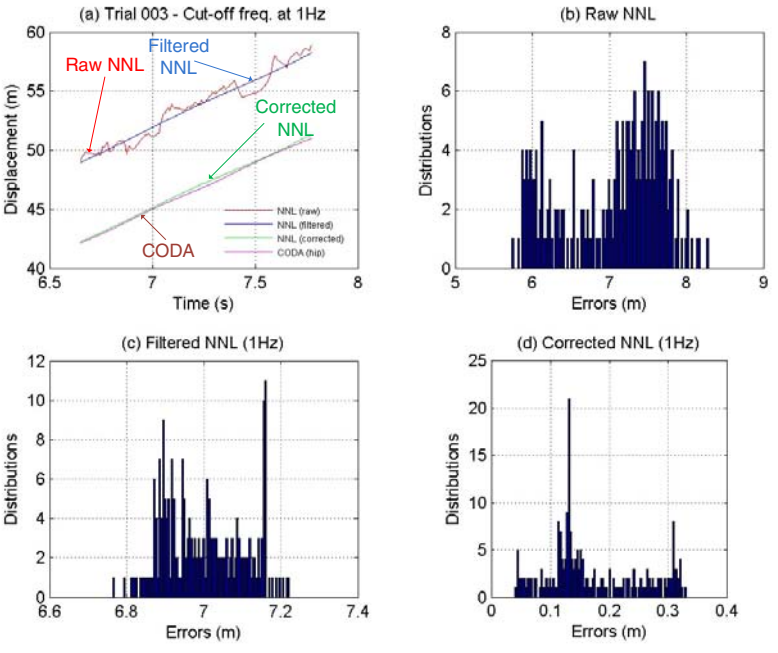


Fig. 5. Error distributions of raw NNL, filtered NNL (1Hz), and corrected NNL vs. CODA (trial 3, set 3)

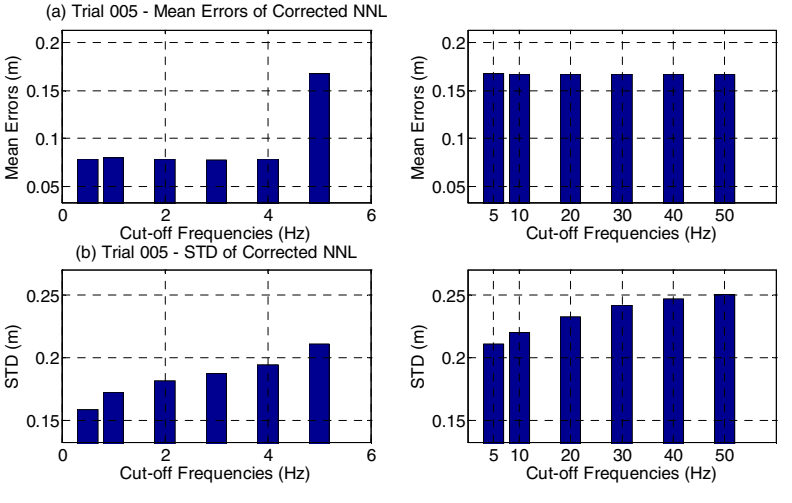


Fig. 6. Mean error and STD at different cut-off frequencies (trial 5, set 2)

track. It was concluded that this bump was caused by the multiple path ground reflection of signals which is associated with the characteristics of the antenna being used.

Mean error and STD were useful to evaluate systematic error and noise. The results (Fig. 6) show that, the lower the cut-off frequency, the smaller the STD, and the mean

error is small and relatively steadily (when cut-off frequency is <5Hz). Also, the estimated trajectory is smoother; whereas at 5Hz, the curve is relatively "wobbly". One may argue that the smaller STD when using a lower cut-off frequency for filtering is a result of the over-smoothing effect, which could lead to small systematic errors; however, a smooth curve with a small STD would be useful for stride length analysis because - for stride length - it is the *difference* between two points that are of interest.

5.2.4 NNL CoM Stride Displacement Analysis

The foot-on times from the FSR system are combined with the corrected NNL data to determine the corresponding CoM stride displacement; the results are compared with the corresponding stride length results from CODA (Table 1). A total of 32 strides were collected. Fig. 7 shows how stride length could be determined from NNL and FSR data: the foot-on timestamps from FSR are determined, the times are used to determine the corresponding forward CoM displacement from NNL data. The result is the CoM stride displacement, which corresponds to the stride length.

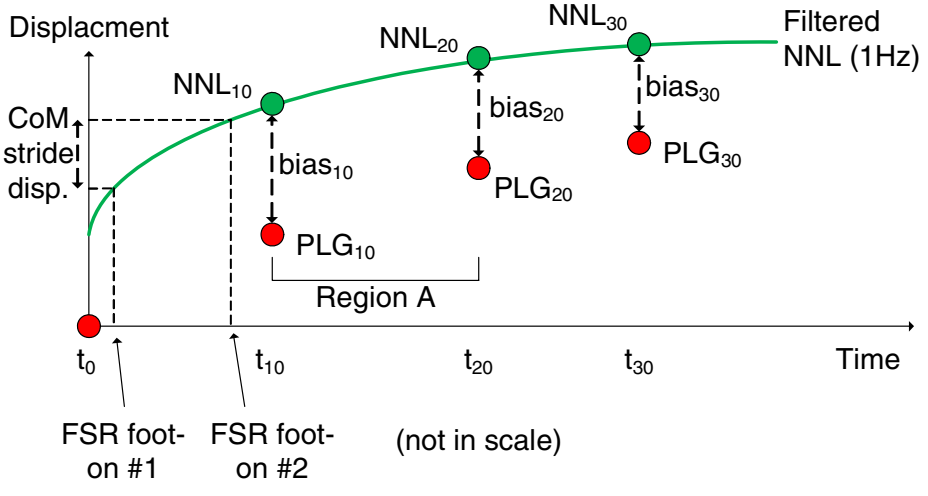


Fig. 7. Stride length determination by combining continuous CoM displacement data from NNL and FSR foot-on data

Table 1. Average mean errors and average STD of NNL "strides" of different sets

Set num	1	2	3
Mean Diff (cm)	5.65	9.2	3.06
STD (cm)	25.16	8.34	14.98

The accuracy of NNL's CoM stride displacement is directly related to the accuracy of the corrected NNL data itself, and it is also related to the relationship between the CoM stride displacement and the actual stride length. The STD of set 1 is slightly higher than the required accuracy, which is in-line with the observations presented in

section 5.2.1: at the acceleration phase, the relationship between stride length and CoM stride displacement (or CoM forward displacement) is relatively weaker than the relationship at later stages. The results suggest that the SESAME IS is capable of determining stride length at a high level of accuracy for the secondary accelerating phase, maximum speed phase, and fatigue phase.

It should be noted that the presented filtering and correction model is an essential element of a practical solution to achieving the highest possible level of accuracy. Averaging raw NNL data over time would not remove the effect of non-constant drift in the data; and it should also be noted that athletes' speed is not constant throughout a sprint (the times in Fig. 4 and Fig. 5 are different, showing that the athletes are running at different speeds during different phases), and as discussed in section 4, the variability of speed during a sprint is subject-dependent. If the non-constant drift is repeatable (i.e. that it is determinable how the system would drift at each specific location on the track), one might argue that one could create a model to address the drift. However, such argument relies on an assumption that the surrounding environment of the track remains unchanged. Such argument cannot be justified because the track is a shared domain, meaning that it is beyond one's control of any future developments, and it is not reasonable to assume resources would be available in the long term to re-calibrate the system every time a change happens. Since any radio-based system that uses a shared radio-band is subjected to interference, the SESAME IS uses the PLG data to minimise the effect of unpredictable events that might affect the accuracy of the system.

6 Applicability and Impact

Since the commissioning of the PLG system at NIAC, the system has been used by all registered coaches at NIAC and their associated athletes during their training sessions in a weekly basis. Recent research work [18][19] have investigated into algorithms for interference-aware radio-based localisation systems however, the impact of such mechanisms on the system's accuracy when deployed in a real environment is not known. Although it is beyond the scope of this paper to investigate interference-free (-aware) radio-based system, the real, raw localisation data from the SESAME NNL system would provide valuable information to carry out evaluation on new correction algorithms.

7 Conclusion

In this paper, the design and experimentation results of the SESAME IS system was presented. The system was designed to determine stride-related information, such as stance time, stride time, and stride length, as well as speed-related info such as split times and continuous location/speed information of high speed running sprinters. The SESAME Integrated System includes a wireless foot-mounted FSR, a radio-based localisation system, and a LG-based split time measuring system. The technologies behind each sub-system, together with practical solutions to integrate and synchronise the heterogeneous sub-systems, are presented. The procedures of the experiments and

the correction model for noisy and non-constant raw radio-based localisation data were presented. Through experimentations and sensor fusion of data from multiple sources, the system achieves an accuracy within 5ms for stance time and stride time measurements; and within ~10cm for stride-related measurements across the secondary acceleration phase, maximum phase, and fatigue phase of a sprint, with a slightly higher variation during the acceleration phase. The applicability and impact of the system are also discussed.

8 Future Work

Part of the future work is to develop custom-made sensor logging boards with much smaller size. The next version of sensor logging boards - which will replace the current NNL tag and the FSR logging board - uses the same NNL AVR chip and an on-board chip antenna. The new board is approximately 2mm thick with half the size of a credit card. The decision to use the same chip (but on a smaller board) is such that one could continue the investigation base on the experiment results presented in this paper.

Another part of the future work involves collecting data from the same integrated system but using a multiple-anchor setup. The idea is that, should one anchor observe the bump; others may not. Thus, even more accurate displacement measurements could be obtained. An initial experiment using four anchors suggest that, Curvilinear Component Analysis (CCA) [17] produces accurate 2D localisation data; 2D localisation would enable the system to monitor athletes running on the oval track.

Acknowledgement

The authors would like to thank Rae Harbird, Alex Atack, Tim Exell, Michelle Manning, Gen Williams, Dawn Tighe, Scott Simpson, David Lease, Ashweeni Beeharee, and Simon Julier for their contributions and support. The authors would also like to thank the athletes who kindly agreed to participate in the studies and experiments. This work was funded by EPSRC grant number EP/D076943.

References

[1] The SEnsing for Sports And Managed Exercise (SESAME) project, http://www.sesame.ucl.ac.uk

[2] Cheng, L., et al.: Analysis of Wireless Inertial Sensing for Athlete Coaching Support. In: Proceedings of IEEE Global Communications Conference (GLOBECOM), New Orleans, USA (December 2008)

[3] Cheng, L., et al.: A Low-cost Accurate Speed Tracking System for Supporting Sprint Coaching. Accepted for publication in the Proceedings of the Institution of Mechanical Engineers, Part P, Journal of Sports Engineering and Technology

[4] Tan, H., Wilson, A.M.: Measurement of stride parameters using a wearable GPS and inertial measurement unit. Journal of Biomechanics 41, 1398–1406 (2008)

[5] nanoLoc Development Kit v1.4, nanotron Technologies, http://www.nanotron.com/EN/PR_nl_dev_kit.php

[6] CODAmotion, http://www.codamotion.com
[7] Charnwood Dynamics Ltd., CODA cx1 User Guide (2006)
[8] Kranz, M., Spiessl, W., Schmidt, A.: Designing Ubiquitous Computing Systems for Sports Equipment. In: Proceedings of IEEE PerCom 2007, pp. 79–86 (2007)
[9] King, R., et al.: Body Sensor Networks for Monitoring Rowing Technique. In: Proceedings of the 6th IEEE International Workshop on Wearable and Implantable Body Sensor Networks, CA, USA (June 2009)
[10] Mann, R.: The Mechanics of Sprinting. CompuSport, Orlando, FL (1990)
[11] Vicon, http://www.vicon.com/products/
[12] Qualisys, http://www.qualisys.com/
[13] Chi, E.: Introducing Wearable Force Sensors in Martial Arts. Pervasive Computing Magazine 04(3), 47–53 (2005)
[14] Courtesy Ferro, A., Rivera, A., Pagola, I., Ferreruela, M., Martín, A., Rocandio, V.: Biomechanical Analysis of the World Championships in Athletics Sevilla'99: 100, 200, 400m sprint events. New Studies in Athletics 16(1/2) (2001)
[15] Baker, J.S., Davis, B.: High intensity exercise assessment: relationships between laboratory and field measures of performance. Journal of Science and Medicine in Sport 5(4), 341–347 (2002)
[16] The gumstix computer, http://www.gumstix.com/
[17] Li, L., Kunz, T.: Localisation Applying an Efficient Neural Network Mapping. In: Proceedings of the 1st International Conference on Autonomic Computing and Communication Systems, Rome, Italy (2007)
[18] Shen, Y., Cai, Y., Xu, X.: Localized Interference-aware and Energy-conserving Topology Control Algorithms. The Proceedings of Wireless Personal Communications: An International Journal 45(1), 103–120 (2008)
[19] Song, B., Lee, H., Chung, K.: Toward A Totally Solving Interference Problem for Ultrasound Localization System. In: The Proceedings of Optical Internet and Next Generation Network (COIN-NGNCON), Jeju, South Korea, July 2006, pp. 162–164 (2006)

Wiselib: A Generic Algorithm Library for Heterogeneous Sensor Networks

Tobias Baumgartner[1], Ioannis Chatzigiannakis[2,3], Sándor Fekete[1], Christos Koninis[2,3], Alexander Kröller[1], and Apostolos Pyrgelis[3]

[1] Braunschweig Institute of Technology, IBR, Algorithms Group, Germany
{t.baumgartner,s.fekete,a.kroeller}@tu-bs.de
[2] Research Academic Computer Technology Institute, Patras, Greece
{ichatz,koninis}@cti.gr
[3] Computer Engineering and Informatics Department, University of Patras, Greece
pyrgelis@ceid.upatras.gr

Abstract. One unfortunate consequence of the success story of wireless sensor networks (WSNs) in separate research communities is an ever-growing gap between theory and practice. Even though there is a increasing number of algorithmic methods for WSNs, the vast majority has never been tried in practice; conversely, many practical challenges are still awaiting efficient algorithmic solutions. The main cause for this discrepancy is the fact that programming sensor nodes still happens at a very technical level. We remedy the situation by introducing *Wiselib*, our algorithm library that allows for simple implementations of algorithms onto a large variety of hardware and software. This is achieved by employing advanced C++ techniques such as templates and inline functions, allowing to write generic code that is resolved and bound at compile time, resulting in virtually no memory or computation overhead at run time.

The Wiselib runs on different host operating systems, such as Contiki, iSense OS, and ScatterWeb. Furthermore, it runs on virtual nodes simulated by Shawn. For any algorithm, the Wiselib provides data structures that suit the specific properties of the target platform. Algorithm code does not contain any platform-specific specializations, allowing a single implementation to run natively on heterogeneous networks.

In this paper, we describe the building blocks of the Wiselib, and analyze the overhead. We demonstrate the effectiveness of our approach by showing how routing algorithms can be implemented. We also report on results from experiments with real sensor-node hardware.

Keywords: Sensor Networks, Algorithms, Library, Heterogeneity.

1 Introduction

Since the initial visions proposed in the SmartDust project [13] ten years ago, Wireless Sensor Networks have seen a tremendous development, both in theory and in practice. On the practical side, we see working sensor networks and applications in many areas, from academia to industrial appliances. There is a large variety of hardware and software to choose from that is easy to set up and use.

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 162–177, 2010.

This success story has also led to a serious practical issue that has not been sufficiently addressed in the past: Sensor node brands are very different in their capabilities. Some nodes have 8-bit microprocessors and tiny amounts of RAM, while others burst with power, being able to run desktop operating systems such as Linux. Consequently, the software running on these systems is very different on the various nodes. While it is easy to write code for a specific platform, it is a very challenging task to develop platform-independent code. Even worse, the operating systems on most sensor nodes provide barely enough functionality to implement simple algorithms. This means that the developer is forced to spend great attention on low-level details, making the process painfully complex and slow.

A parallel success story can be observed on the theoretical side, where the development of distributed algorithms for many actual or hypothetical problems has grown into a research field of its own. This has led to a large variety of highly sophisticated algorithms for all kinds of tasks. Unfortunately, many of them have never been tried in practice, due to the overly difficult implementation process. Where algorithms are implemented, they are hard to share and compare, as implementations cannot be easily ported to new platforms. Moreover, many important challenges are not even addressed, as they can only be identified and resolved by close collaboration between theory and practice.

This growing gap between theory and practice forms a major impediment for exploiting the possibilities of complex distributed systems. The *Wiselib* is our proposal to remedy this unfortunate situation. We present a framework, written in C++, for platform-independent algorithm development. Each algorithm written for the Wiselib can be compiled for any supported system without changing any line of code. It provides simple interfaces to the algorithm developer, with a unified API and ready-to-use data structure implementations. The *Wiselib* addresses the following issues:

Platform independence. Wiselib code can be compiled on a number of different hardware platforms, usually without platform-dependent configurations, i.e., no "`#ifdef`" constructions. See Section 3.1 for details.

OS independence. Wiselib code can be compiled for different operating systems. This includes systems based on C like Contiki, as well as C++ (the iSense firmware) and nesC (TinyOS).

Exchangeability. Algorithms and applications can be composed of different components that interact using well-defined interfaces, called *concepts*. Components can be exchanged with other implementations without affecting the remaining code. Moreover, both generic components and highly optimized platform-specific components can be used simultaneously.

Broad algorithm coverage. The Wiselib currently covers a large variety of algorithms. It will contain algorithms for each of the following categories:

1. routing algorithms
2. clustering algorithms,
3. time-synchronization algorithms,
4. localization algorithms,
5. data dissemination, and
6. target tracking.

Cross-layer algorithms. In Wiselib an algorithm can be designed to use other algorithm concepts, thus enabling the use of existing algorithms for the implementation of more complex ones. Moreover, we can stack protocols on top of each other, extending their functionality. See Section 5 for details.

Standard compliance. The library is written in a well-defined language subset of ISO C++. This has a number of benefits over custom languages such as nesC: The compilers are more mature and better supported, and there is a large user base that knows C++ from desktop development.

Scalability and efficiency. The Wiselib is capable of running on a great variety of hardware platforms, with CPUs ranging from 8-bit microcontrollers to 32-bit RISC CPUs, and with memory ranging from a few kilobytes to several megabytes. Algorithms need to be very resource-friendly on the platforms from the lower end, and at the same time be able to use more resources if available.

To our knowledge, the Wiselib is the only successful attempt to achieve all of these goals at once. In this paper, we present the basic building-blocks of the Wiselib, and show that the flexibility of the design has barely any overhead—neither in code size nor in run-time; one can simply add new algorithms only by following the presented approach using the Wiselib interfaces. The algorithm can then run on each supported sensor node or simulation platform. Our goal is to achieve a state in which such an algorithm runs on heterogeneous sensor networks, and even more, networks in which some parts consist of virtual nodes running in a simulator.

This paper is organized as follows: The next section provides an overview of related work, covering competing approaches as well as implementations that inspired this work. Section 3 explores the problem space by discussing the target platforms on which we wish to run the Wiselib. Section 4 presents details on the design of the Wiselib. In Section 5 we describe example implementations of routing algorithms; in Section 6, we report on the surprisingly small code and memory footprint on different platforms. Section 7 describes the current distribution of the Wiselib. We conclude the paper in Section 8.

2 Related Work

Efficient algorithm libraries have a long-standing tradition on desktops and servers. The three libraries that motivated our work are the Standard Template Library (STL), the Computational Geometry Algorithms Library (CGAL) [4], and Boost [2]. They share a great programming concept that we heavily use for the Wiselib: Using C++ templates, one can construct complex object-oriented software architectures that can be parameterized for many different applications. The price of generality is paid at compile time. The final binary contains highly efficient and specialized code, so that there is no overhead at runtime.

The situation in sensor networks is not as promising. There have been approaches to overcome the issues of incompatible nodes by providing generic operating systems that run on multiple platforms. Examples are Contiki [6] and

TinyOS [20]. Neither runs on all platforms we are envisioning for the Wiselib. Even worse, both introduce new programming paradigms that are valid only for the specific targets, such as protothreads in Contiki, and the whole programming language nesC [7] of TinyOS. The C-inspired nesC attempts to allow for the construction of component architectures with early binding, similar to the Wiselib, but achieves this through introducing a new language that requires a custom compiler.

A challenging issue are heterogeneous networks. It is very simple to have nodes exchange messages if they are of the same kind, and with the same operating systems. It becomes surprisingly hard to let nodes of different brands communicate with each other, even if both of them use standardized IEEE 802.15.4 radios. A promising approach is the Rime Stack [10,5], a layered communication stack for sensor networks. It runs only on Contiki. Recently, Sauter et al. [16] demonstrated that is is possible to communicate between sensor nodes running Contiki and TinyOS. Since TinyOS uses IEEE 802.15.4, the Rime Stack and Chameleon Module had been modified on Contiki.

Another attempt to produce a well-defined environment that runs on different platforms was proposed by Boulis et al. [3]: SensorWare defines a custom scripting language; its syntax is based on Tcl. Consequently it focuses on richer platforms with at least 1 Mbyte of ROM and 128 KBytes of RAM. A similar approach is Maté [14], a virtual machine running on top of TinyOS. It targets also small devices with a very limited amount of resources, using a custom assembler-like language.

Not surprisingly, there are are also attempts to run a Java Virtual Machine (JVM) on sensor nodes [17]. Squawk [18] is a JVM by Sun Microsystems that runs on Sun Spots. Obviously such an approach is not suited for low-end sensor nodes, and also not for time-critical algorithms.

A different approach are macroprogramming frameworks such as Kairos [9], Marionette [22], and MacroLab [11]. Instead of writing code for individual nodes, the whole network is addressed with a single program. This is generally achieved by providing a script language that is executed automatically on all nodes, without the need for reprogramming any node in the network.

3 Problem Space

3.1 Heterogeneity

When developing an algorithm library for sensor networks, one must deal with a great variety of different hardware and software platforms. Table 1 shows an overview of platforms that were taken into account for the development of the Wiselib.

The operating systems vary from system-specific implementations such as iSense and ScatterWeb to generic approaches such as Contiki, TinyOS, and Linux. The preferred programming languages vary with the OSs. The iSense firmware has been developed in C++, whereas the ScatterWeb firmware uses plain C. TinyOS uses a custom language, the C extension *nesC* [7]. Support

Table 1. Evaluation of potential target platforms. The columns refer to the type of microcontroller, the standard operating system, the programming language for it, what kind of dynamic memory is available, the amount of ROM and RAM, and the bit width.

Hardware	Firmware/OS	CPU	Language	Dyn Mem	ROM	RAM	Bits
iSense	iSense-FW	Jennic	C++	Physical	128kB	92kB	32
ScatterWeb MSB	SCW-FW	MSP430	C	None	48kB	10kB	16
ScatterWeb ESB	SCW-FW	MSP430	C	None	60kB	2kB	16
Tmote Sky	Contiki	MSP430	C	Physical	48kB	10kB	16
MicaZ	Contiki	ATMega128L	C	Physical	128kB	4kB	8
TNOde	TinyOS	ATMega128L	nesC	Physical	128kB	4kB	8
iMote2	TinyOS	Intel XScale	nesC	Physical	32MB	32MB	32
GumStix	Emb. Linux	Intel XScale	C	Virtual	16MB	64MB	32
Desktop PC	Shawn	various	C++	Virtual	unlimited	unlimited	32/64
Desktop PC	TOSSIM	(ATMega128L)	nesC	(Physical)	unlimited	unlimited	(8)

for dynamic memory, `malloc()` and `free()`, is only available for some systems. Using the ScatterWeb firmware, the size of all memory blocks must be known at compile time, whereas the iSense firmware provides a full implementation for the C++ operators `new` and `delete`. This is done with the aid of an own memory allocation implementation. Similar approaches are provided by TinyOS via *TinyAlloc*, and Contiki via the *managed memory allocator* or *memb block memory allocator*. Only the Linux-based node supports virtual address space for processes. There are also significant differences in the amount of available memory, ranging from a few kilobytes to 64 MByte in the GumStix. Finally, we must also deal with different bit widths. The Atmel Atmegas are 8-bit microcontrollers, the MSP430 are 16-bit microcontrollers, whereas the rest are 32-bit microcontrollers. There are a number of challenges stemming from the nodes' properties and capabilities. These became additional library requirements.

Limited Memory. The algorithms may run on tiny microcontrollers for which the provided memory is very limited. On the one hand, this affects the ROM. The generated code for an algorithm must be as small as possible to fit into memory. On the other hand, the RAM is affected. Routing tables, for example, cannot be arbitrarily long so as not to exhaust the limited main memory. Additionally, the node representation that is used for storing the neighborhood must be as small as possible, but must also meet the demands of the used algorithms. At the same time, when running on a node with plenty of memory, performance gains can and should be achieved by employing more advanced data structures.

Physical Dynamic Memory. The availability of dynamic memory allocation is already a big step forward, allowing for efficient data structures. However, most implementations only provide physical addresses, and some are even unable to join adjacent freed memory blocks. Shifting of pages to join free blocks is impossible on all nodes with physical memory. Even a simple vector implementation with $O(\log n)$ amortized insertion time would leave behind a trail of $O(\log n)$ free blocks of various sizes. Therefore, data structures must be carefully re-analyzed to take these special considerations into account.

Limited Computation Power. Because algorithms may run on small microcontrollers, efficiency plays an essential role. Examples are message reception in

an interrupt or iterating over a neighbor table to select the next routing node. This also constrains the Wiselib not to enforce the use of slow operations (such as excessive pointer indirection) through the provided framework.

Compiler Variance. Our library must run on multiple hardware platforms. Different compiler versions must be supported, so it is important that only standard features of the selected programming language are used.

Data Access. When accessing data at arbitrary locations in memory, alignment problems can occur. For example, a cast of a 16bit integer works for both MSP430 and Jennic, when it starts at an even address. But when it starts at an odd address, it fails on both platforms. However, a cast of a 32bit integer works on all even addresses on a MSP430, but for Jennic only on quad-byte boundaries.

Moreover, when exchanging data in heterogeneous systems, the byte order must be taken into account, because some systems are big endian, whereas others are little endian.

3.2 C++ in Embedded Systems

The Wiselib must cover all of the previously mentioned hardware and software platforms; the latter are developed in different programming languages. Hence, an appropriate programming language must be found. We chose C++ [19], because it combines modern programming techniques with the ability of writing efficient and performant software. The use of C++ in embedded systems has already been evaluated [12]. Based on this report and own evaluations, we selected a subset of the language to be used in the Wiselib.

C++ allows modern OO designs. Object-Oriented programming is standard on the desktop for quite some time by now, and has proven to ease the development of complex systems. Moreover, C++ is a fully typesafe language. This speeds up the development process, as it catches type errors at compile time. Given the tediousness of debugging on sensor nodes, this is a huge achievement.

The most important language feature for the Wiselib are templates [21,1]. Templates can be used to develop very efficient and flexible applications. The basic functionality of templates is to allow the use of generic code that is fully resolved by the compiler when specific types are given. Thereby, only the code that is actually needed is generated, and methods and parameters as template parameter can be accessed directly. We use the well-established technique of template-based "concepts" and "models", where the former are not specified as actual code, but rather as formal specifications in documentation. It lists the required and provided types, as well as member function signatures. Models are implementations of concepts, using template specializations, without any inherent runtime overhead. Both concepts and models allow for polymorphism, including multiple inheritance. These techniques are used successfully in standard C++ libraries, such as the STL, Boost [2], and CGAL [4]. The Wiselib employs these methods in the same manner, i.e., using standard compiler features without custom additions.

Another basic feature in C++ is virtual inheritance. When declaring a method as `virtual`, the compiler has to generate a vtable consisting of function pointers

Table 2. Availability of C++ compilers for selected platforms

Architecture	Compiler	Binary	Base	libstdc++	Basic C++	Syntax Templates
Jennic	ba-elf-g++	√	GCC 4.2.1	√	√	√
MSP430	msp430-g++	-	GCC 3.2.3	-	√	√
ATMega128L	avr-g++	-	GCC 4.1.2	-	√	√
Intel XScale	xscale-g++	√	GCC 3.3.1	√	√	√

to the appropriate methods. Whenever such a method is called, it has to be looked up in the `vtable` first, thereby requiring pointer indirection. This leads to an increase of both program memory and run-time, and makes some compiler optimizations impossible. Hence, we do not use virtual inheritance in the Wiselib. We substitute this feature by templates.

Two more features that are not used in the Wiselib are run-time type information (RTTI) and exceptions. Both result in significant runtime and code-size overhead, as already shown in [12].

There are C++ compilers available for all of our target platforms. See Table 2 for an overview. Some platforms lack support for libstd++, which includes the operators `new` and `delete`. The STL is also not available everywhere. All compiler support the C++ features we build upon, i.e., template and member specializations.

All compilers are based on GCC, and thus there are no considered drawbacks from compiler incompatibilities. There are some minor limitations due to the missing libstdc++ on some systems, which have no impact on the Wiselib.

4 The Wiselib

The core design pattern for the Wiselib are generic programming techniques that are implemented using C++ templates. The basic idea is to pass the important functionality as template parameters to an algorithm: implementations of OS specific code, and data structures. Hence, it is possible to compile an algorithm exactly for the current needs.

4.1 Architecture

The fundamental design principle of the Wiselib consists of concepts and models, which have already been discussed in Section 3.2. We feature an architecture with three main pieces: algorithms, OS facets, and data structures. The idea is shown in Fig. 1.

First of all, there are concepts for algorithms. There is one concept per category, whereby a category groups algorithms by their basic functionality, e.g. routing or localization. Any algorithm model implements one or multiple concepts, and is basically a template expecting various parameters. These parameters can be both OS facets and data structures.

OS facets represent the connection to the underlying operating system or firmware—for example, concepts for a radio or timer interface. Thus, the facets

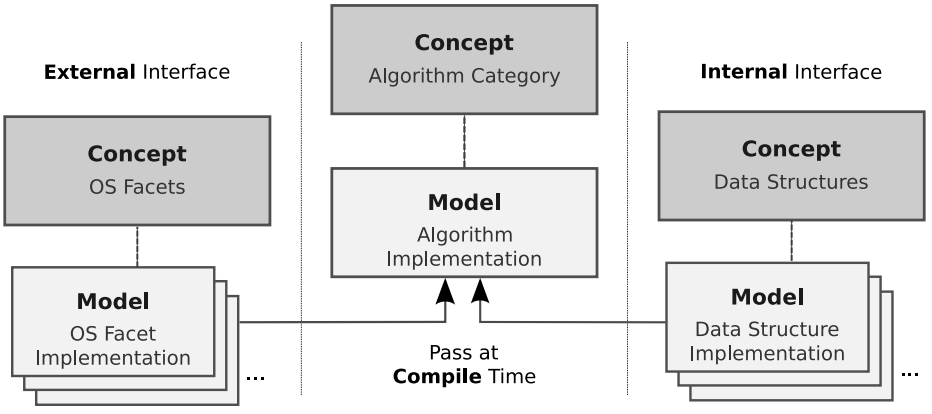


Fig. 1. Wiselib Architecture

provide a lightweight abstraction layer to the OS. Note that the facets are merely type definitions and wrapper functions, they are supposed to contain no replication of OS functionality.

With the aid of data structures, an algorithm can scale to the platform it is compiled for. For instance, static data structures can be passed on tiny platforms without dynamic memory management, whereas highly dynamic and efficient data structures are passed on powerful microcontrollers or desktop PCs.

4.2 External Interface

The "external interface", consisting of OS facets, represents the connection to the underlying OS. Implementations of these facets are passed to an algorithm as template arguments. The compiler should mostly be able to directly resolve such calls to the OS. For example, when registering a timer can be done using one line of code, it is implemented as an inline function in the appropriate timer model. Hence, the result would be a direct call to the OS function, and thus there would be no overhead, neither in code size nor in execution time. In C-based operating systems (we see TinyOS in this group), the OS facets have to provide a translation between C++ member function calls and C function calls, and they have to convert C++ members to C callback pointers. This is where an actual price of generality has to be paid. Fortunately, as we report in Section 6, this price is very low.

Several models of the same concept for an OS facet can also be made available, each with its own advantages for special purposes. The user can pass the best available model to an algorithm at compile time, without extra overhead.

An example for a model of the OS facet "radio" is as follows. It is for the C++-based iSense firmware:

```
1 template<...> class iSenseRadioModel {
2 static int send(Os *os, id_t id, size_t len, data_t *data)
3 { os->radio().send( id, len, data, 0, 0 ); }
```

The example shows the implementation of a simple send method offered by a radio model. Since it is only one function call, it can be directly resolved by the compiler without generating any overhead.

Concept Inheritance. The above example of the radio's send() method with destination address and payload is defined in the basic radio concept. Routing algorithms, for example, which do only need to send and receive messages without any further information such as RSSI values, or requirements such as reliable delivery can use implementations of this concept.

We also allow for concept inheritance, so that the basic radio concept can easily be extended. If an algorithm needs access to RSSI (or LQI) values, a derived concept can be used. It extends the basic one with a receive method that provides additional values.

Stackability. A major design aspect for the radio concept is stackability, i.e., the possibility to build a layered structure of multiple radios. The topmost layer is not aware to which and how many layers it is connected. The big advantage of this approach is that we can build a "virtual radio" that runs on top of a radio model, and is passed to an algorithm in its radio template parameter. Doing so, we can easily implement an algorithm for heterogeneous sensor networks. It is even possible to communicate between nodes that use different kinds of node IDs—because the virtual radio hides the real node addresses and provides, e.g., generic 128 bit addresses.

Another possibility is to hide a complete routing algorithm behind an OS facet. For example, when writing out debug messages, this happens generally to the UART. But by passing another model, we can forward debug messages over a routing algorithm to a gateway, where all these messages are collected. The topmost algorithm does not need to be aware of the model it works on—it must only use the appropriate concept.

Message Delivery in Heterogeneous Systems. Another problem that is addressed using our software design is message delivery in heterogeneous networks. There are basically two problems that occur: different byte-order, and differences in alignment handling. Byte order issues are solved by sticking to network byte order in messages. Alignment is addressed via template specialization. We provide a serialization class that provides generic `read` and `write` methods for all data types.

4.3 pSTL

Not all of our target systems provide dynamic memory allocation. To our knowledge, no variant of the STL fulfills our requirements: not using libstdc++, new/delete, exceptions, and RTTI.

Consequently, we provide the pSTL, an implementation of parts of the STL that does neither use dynamic memory allocation nor exceptions nor RTTI. We ensure that each of the provided data structures works on each supported hardware platform. At the moment, implementations for `map`, `vector`, and `list` are available. Naturally, the pSTL will grow with increasing demand.

4.4 pMP

For many tasks in embedded systems, multi-precision arithmetic is needed, e.g. for cryptographic and data aggregation purposes. Currently there exist a number of software libraries that implement big-number operations, e.g., gnuMP [8]. Such libraries heavily rely on dynamic memory allocation to represent big-numbers and carry out the operations. Moreover, to achieve performance speedups, highly optimized assembly code is used, taking advantage of specific hardware instructions. Unfortunately, the hardware types used in WSN platforms (e.g., ATMEGA, Jennic) support neither dynamic memory allocation nor the specific hardware instructions used by gnuMP and other libraries. Hence it is very difficult to port such implementations to our platforms, if not impossible at all.

Therefore, we provide the pMP, an C-based implementation of big-number operations that does not use dynamic memory allocation. Of course such a library cannot be compared in terms of efficiency with gnuMP, but it is the only one available currently. In particular, it implements some basic operations like xor, shiftleft and modulo multiplication operations which are required for elliptic curve cryptography. It is certain that the pMP will grow regarding future needs.

4.5 Algorithm Support

The central piece of the Wiselib are the algorithms. They are grouped into categories, see Section 1. Algorithm implementation can belong to several categories, which is common for cross-layer algorithms.

Each algorithm class consists of a concept for the algorithm itself, and some concepts for the data structures that are typically necessary for this class. This decouples the algorithm logic, which is invariant over different platforms, from data storage, which heavily changes when an algorithm is ported to a platform of different characteristics.

The benefit of having a well-defined algorithm interface is that algorithms are easily interchanged for testing purposes, ideally this is done by simply altering a class name in the initialization code. The second—much more important—benefit is that an algorithm developer can start coding by copy-and-paste, instead of having to go through a design phase. Such a design phase can be quite lengthy, if the goal is to achieve maximal portability. Until now, theoreticians wishing to evaluate high-level algorithms often found it hard to develop for embedded devices: this lowers the bar considerably.

Providing a diverse set of data structure implementations serves the goal of scalability: For each data structure, e.g., routing tables, neighborhood cluster maps, and position maps, a set of implementations matching the span of platforms is provided. For low-end architectures such as the MSP430, structures are needed that use static storage whose size is known at compile-time. Such structures will inevitably be inefficient in terms of runtime. For high-end architectures using Xscale processors or simulation environments, highly optimized data structures with dynamic memory management and huge memory overhead can be employed, resulting in high efficiency. It is even feasible to utilize the

STL. The choice of data structures has no impact on the algorithm code, and can simply be configured at algorithm initialization. This results in algorithms that not only scale down to very limited devices, but also scale up to powerful nodes, utilizing all the available resources on them.

5 Case Study: Secure Routing Algorithms

We show the benefits of C++ and template-based design by presenting two examples: routing and cryptography algorithms. First we present either of the approaches as a single concept. Then we show how easily individual implementations can be combined to generate secure routing algorithms.

Routing Algorithms. When designing a concept for an algorithm class, one wishes to cover all kinds of special case, while staying as generic as possible. This is because each method in the concept must be implemented by each model. Hence, our concept for a routing algorithm consists of only six methods.

First, we need a method for setting the pointer to the OsModel that is needed when calling static member functions from the External Interface. Then we have two methods for enabling and disabling the routing algorithm, which is useful when the routing should only be run in certain points in time, for example for energy-saving issues. Next, a potential user of the routing algorithm must be able to register and unregister a callback for message reception. At last, there is the method for sending messages to other nodes in the network. The Routing Concepts specializes the Radio Concept, so that routing algorithms can be used as virtual radio interfaces for other algorithms. The concept looks as follows:

```
1 concept Routing {
2   void set_os(OsModel* os);
3   void enable(void);
4   void disable(void);
5   void send(node_id_t receiver, size_t len, data_t* data);
6   template <class Callee, void (Callee::*Method)
7                           (node_id_t, size_t, data_t*)>
8   int reg_recv_callback(T *obj_pnt);
9   void unreg_recv_callback(int);
10 };
```

Cryptography. Adapting cryptographic algorithms to embedded systems is a difficult task due to resource limitations. Unlike the routing case, we avoid covering all special cases of crypto algorithms. We provide a simple concept with algorithm implementations that will be viable solutions for the tiny sensors.

Our generic concept for a crypto algorithm consists of five methods. We provide methods for key setup, encryption and decryption of data blocks. The concept looks as follows:

```
1 concept Crypto {
2   void set_os(OsModel* os);
3   void enable(void);
4   void disable(void);
5   void key_setup(node_id_t, data_t* key);
6   void encrypt(data_t* in, data_t* out, size_t length);
7   void decrypt(data_t* in, data_t* out, size_t length);
8 };
```

Secure Routing. In this section, we describe how the individual routing and cryptographic implementations can be combined to result in secure routing algorithms. Note that any available routing implementation can be combined with any available crypto algorithm without a single change in their code.

We therefore implement the routing concept, and accept a routing algorithm and a crypto algorithm as template parameters. Internally, we only use the passed types. For example, when the secure routing is enabled, it in turn enables the routing and crypto algorithm. When a message is sent, it first encrypts the passed bytes, and then passes the encrypted data to the routing algorithm. Then, when a message is received at the destination, it is first decrypted, and then passed to the registered receivers. The secure routing looks then as follows:

```
1 template<typename Routing,
2          typename Crypto>
3 class SecureRouting {
4    void set_os(OsModel* os);
5    [...] // all methods described in the routing concept
6    void unreg_recv_callback(int);
7    Routing routing_;
8    Crypto crypto_;
9 };
```

Since it implements the routing concept, it can be passed and used by any application that deal with routing algorithms. However, the process of both encryption and decryption is completely transparent.

6 Experimental Results

In order to demonstrate the efficiency of our generic approach, we ran different experiments on supported platforms. We evaluated two main parts of the Wiselib: First, the overhead of the connection to the underlying OS; second, properties of implementations of a first set of algorithms.

6.1 External Interface

We tested the performance of Wiselib system calls compared to native OS calls on three different platforms. The results are shown in Table 3.

OS calls that are short enough to be directly inlined by the compiler, such as sending a message on iSense platforms or reading the node ID in Contiki do not have any overhead. However, other parts in the OS connection produce a small overhead due to an additional layer of indirection. This is mainly because of incompatibilities between C function pointers and C++ member function pointers, and a required translation between them. But as shown in the performance

Table 3. Performance costs of Wiselib calls compared to native OS calls

	iSense			Contiki			ScatterWeb		
	Native	Wiselib	Cost	Native	Wiselib	Cost	Native	Wiselib	Cost
Read ID	$2\mu s$	$2\mu s$	0%	$<1\mu s$	$<1\mu s$	0%	$<1\mu s$	$<1\mu s$	0%
Send Message	$282\mu s$	$282\mu s$	0%	$336\mu s$	$345\mu s$	3%	$898\mu s$	$921\mu s$	3%
Set Timer	$135\mu s$	$141\mu s$	4%	$77\mu s$	$100\mu s$	30%	$20\mu s$	$43\mu s$	115%

Table 4. Code-size overhead of OS facets. Shown is ROM (.text) and RAM (.bss + .data) in bytes.

	iSense	Contiki	ScatterWeb
Radio	856+240	428+ 72	316+ 40
Timer	868+240	352+210	270+ 80

evaluation, this overhead is very small—if at all, then only in terms of microseconds. Similar delays would also be produced by alternative approaches, but by using C++ and templates the compiler is able to remove this overhead wherever reasonable. This is possible due to the implicit inline declaration of methods.

Time efficiency is only one performance measure; the other is code space. We evaluated the needed size for the two OS facets radio and timer for different platforms. The results are shown in Table 4.

Because the concepts for radio and timer were kept simple, each implementation required at most a few hundred lines of code. This led not only to a slight structure, but also enhanced maintenance issues. In addition, even the integration of a completely new platform can be done without too much effort.

Especially the facets for the ScatterWeb platform show a small amount of overhead of less than 600 bytes in ROM, and 120 bytes in RAM. Even the 1.7kB of iSense are tolerable, since it is a 32bit-platform with corresponding overhead in machine language instructions.

An important factor when estimating the code-size overhead is that it is constant, and thus do not grow with the integration of further algorithms. The interfaces also provide a powerful abstraction of the underlying OS, facilitating implementations of many additional algorithm categories.

6.2 Algorithms

We implemented different algorithms for the routing concept: DSDV, DSR, a simple tree routing, and a flooding algorithm. Each algorithm has been compiled for, and tested on each supported platform. Table 5 shows the resulting code sizes and initial RAM usage for the several platforms.

Table 5. Evaluation of code size as ROM size (.text) and RAM size (.bss + .data) in bytes.

	16-bit OS		32-bit OS	Simulators	
Algorithm	Contiki	ScatterWeb	iSense	Shawn	TOSSIM
DSDV	1446+ 72	1466+ 72	4776+136	4351+ 4	19146+ 4
DSR	1964+338	1716+238	5396+356	6918+ 4	20845+ 4
Tree	920+ 16	724+ 14	4060+ 24	2974+ 4	9946+ 4
Flooding	1122+ 50	762+ 34	2864+ 68	2260+ 4	10192+ 4

It is clearly visible that our algorithm implementation perfectly fits into the target platforms, as the impact of the generality of the code is very low, in terms of both code and memory. However, the given code sizes show only the pure demand of the algorithm—without considering the external interface.

Table 6. Stack latency in Wiselib (measured on the iSense devices)

	Dummy Routing	Dummy Routing, Dummy Crypto	DSDV Routing	DSDV Routing, Dummy Crypto
Latency	6.08 msec	6.09 msec	6.72 msec	6.75 msec

Table 7. Comparison between Wiselib and TinyECC, for encryption/decryption runtime

	TinyECC optimized		TinyECC		Wiselib	
Hardware	Encrypt	Decrypt	Encrypt	Decrypt	Encrypt	Decrypt
TelosB	6.53sec	4.25sec	84.9sec	42.73sec	114.78sec	56.02sec
MicaZ	3.9sec	2.6sec	61.4sec	31.87sec	118.4sec	57.84sec
Tmote Sky	3.27sec	2.12 sec	42.55sec	21.41sec	115.98sec	56.91sec
iSense	-	-	-	-	22.9sec	11.84sec
ScatterWeb	-	-	-	-	102.93sec	50.42sec

Each of the routing models can also be combined with a crypto algorithm—as shown in Section 5. The first point of interest is the overhead of multiple layers of algorithms are. We estimated the average latency by the Wiselib layers. The experiments were held on the iSense platform. The latency was measured as the average of 200 message exchanges: a) through a dummy routing algorithm and a dummy routing algorithm combined with a dummy crypto algorithm and b) through a DSDV routing algorithm and a DSDV routing algorithm combined with a dummy crypto algorithm. We conclude that stack latency overhead is minimal, as shown in Table 6.

As a second experiment regarding the combination of routing and crypto algorithms, we estimated the run-time of a crypto algorithm (Elliptic Curve Integrated Encryption Scheme) through Wiselib for various platforms, and we compared it with that of TinyECC[15] in Table 7. We did not focus on optimizing the code; that is why TinyECC runtime is generally faster. However, our algorithm can be executed on a variety of platforms.

Also, with the aid of template specializations—as also used in message delivery—code can be optimized and adapted for certain platforms. Depending on the compilation process, the compiler can select exactly the code that fits best for the current platform. For example, when an algorithm is compiled for iSense, the AES hardware could be used for the crypto routines.

7 Accessing the Wiselib

There are different demands for the users of the Wiselib. *Application developers* are interested in stable algorithms that were thoroughly tested for all supported platforms. They do not contribute own implementations to the Wiselib; instead, they only integrate existing algorithms in their applications. *Algorithm developers* on the other hand contribute code to the Wiselib. Algorithms may be under development and can not be ensured to run on each platform.

We therefore provide two distributions: *Stable* and *Testing*. The former contains only algorithms that were run through different tests, particularly for each supported platform. Concepts that are implemented for the stable distribution

are also expected not to be changed anymore, if not strongly needed. In contrast, the testing distribution contains newly implemented algorithms. They may not be tested on each platform—in particular since not each algorithm developer has each platform available. This can also lead to changes in concepts, when it is noticed that not all platforms can be covered satisfactorily. In general, the objective here is to release early, and release often.

The Wiselib can be accessed under http://wisebed.eu/wiselib. There is a Wiki available that contains documentation. In addition, there is also a Trac running to report software bugs and collect suggestions for improvement.

8 Conclusion and Future Work

In this paper, we have introduced our generic algorithm library for wireless sensor nodes, the Wiselib. It is aimed at allowing algorithm researchers to quickly implement distributed algorithms on actual sensor nodes. The implementation process requires no deep understanding of the target platform, as the library provides a unified API that abstracts the technical details. Unlike all other approaches with the same goal, or at least the ones we are aware of, Wiselib algorithms suffer next to no runtime or memory overhead from the generality.

The Wiselib is written in standard ISO C++, using advanced OO techniques to encapsulate the operating system and to allow complex OO architectures that can be fully resolved by an optimizing compiler. Specifically, the Wiselib makes heavy use of templates, as they are resolved at compile time, leaving no binding efforts to runtime. Certainly, generality does not allow to provide highly optimized code. Fortunately, our open design allows to provide such hardware specific optimizations without hindering the generality of the algorithm implementation. This is extremely important since algorithm development can be decoupled from application development where platform specific optimizations are performed.

We demonstrate the effectiveness of the Wiselib by implementing a number of routing algorithms and cryptography algorithms. We show that the produced code is very lean and it works on a large variety of sensor platforms. The library allows us to easily stack different types algorithms with almost zero overhead. We build upon this feature and demonstrate the ability to interchange algorithms without affecting the operation of other algorithms at different stack level. These features essentially provide endless possibilities to application developers as more algorithms and algorithmic concepts are introduced in Wiselib.

We expect the Wiselib to grow much beyond the current state, and to become a standard tool for WSNs in the near future. We also wish to look into other categories of algorithms such as MAC layer protocols, energy saving schemes and topology control protocols.

Acknowledgement. This work has been partially supported by the European Union under contract number ICT-2008-224460 (WISEBED).

References

1. Alexandrescu, A.: Modern C++ Design. Addison-Wesley, Reading (2001)
2. Boost, http://www.boost.org

3. Boulis, A., Han, C.-C., Srivastava, M.B.: Design and implementation of a framework for efficient and programmable sensor networks. In: Proceedings of MobiSys 2003, pp. 187–200. ACM, New York (2003)
4. CGAL: Computational Geometry Algorithms Library, http://www.cgal.org
5. Dunkels, A.: Poster abstract: Rime – a lightweight layered communication stack for sensor networks. In: Proceedings of EWSN 2007, Poster/Demo session (2007)
6. Dunkels, A., Gronvall, B., Voigt, T.: Contiki - a lightweight and flexible operating system for tiny networked sensors. In: LCN 2004: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (2004)
7. Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., Culler, D.: The nesc language: A holistic approach to networked embedded systems. In: Proceedings of Programming Language Design and Implementation, PLDI (2003)
8. GNUMP: GNU Multiple Precision Arithmetic Library, http://gmplib.org/
9. Gummadi, R., Gnawali, O., Govindan, R.: Macro-programming wireless sensor networks using kairos. In: Prasanna, V.K., Iyengar, S.S., Spirakis, P.G., Welsh, M. (eds.) DCOSS 2005. LNCS, vol. 3560, pp. 126–140. Springer, Heidelberg (2005)
10. He, Z., Österlind, F., Dunkels, A.: An adaptive communication architecture for wireless sensor networks. In: Proceedings of ACM SenSys (2007)
11. Hnat, T.W., Sookoor, T.I., Hooimeijer, P., Weimer, W., Whitehouse, K.: Macrolab: a vector-based macroprogramming framework for cyber-physical systems. In: Processings of the ACM SenSys 2008, New York, NY, USA, pp. 225–238 (2008)
12. ISO/IEC JTC1 SC22 WG21. ISO/IEC TR 18015: Technical Report on C++ Performance. Technical report (February 2006)
13. Kahn, J.M., Katz, R.H., Pister, K.S.J.: Next century challenges: mobile networking for "smart dust". In: MobiCom 1999: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, pp. 271–278. ACM, New York (1999)
14. Levis, P., Culler, D.: Mate: A tiny virtual machine for sensor networks. In: International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, CA, USA (October 2002)
15. Liu, A., Ning, P.: TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. In: IPSN 2008: Proceedings of the 7th international conference on Information processing in sensor networks (2008)
16. Sauter, R., Marrón, P.J., Dunkels, A., Voigt, T., Tsiftes, N., Finne, N., Österlind, F., Eriksson, J.: Demo abstract: Towards interoperability testing for wireless sensor networks with cooja/mspsim. In: Roedig, U., Sreenan, C.J. (eds.) EWSN 2009. LNCS, vol. 5432. Springer, Heidelberg (2009)
17. Shaylor, N., Simon, D.N., Bush, W.R.: A java virtual machine architecture for very small devices. In: LCTES 2003: Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems (2003)
18. Simon, D., Cifuentes, C.: The squawk virtual machine: Java on the bare metal. In: OOPSLA 2005, pp. 150–151. ACM, New York (2005)
19. Stroustrup, B.: The C++ Programming Language. Addison-Wesley, Reading (2000)
20. TinyOS, http://www.tinyos.net
21. Vandevoorde, D., Josuttis, N.M.: C++ Templates: The Complete Guide. Addison-Wesley, Reading (2003)
22. Whitehouse, K., Tolle, G., Taneja, J., Sharp, C., Kim, S., Jeong, J., Hui, J., Dutta, P., Culler, D.: Marionette: using rpc for interactive development and debugging of wireless embedded networks. In: IPSN 2006, New York, USA, pp. 416–423 (2006)

Selective Reprogramming of Mobile Sensor Networks through Social Community Detection

Bence Pásztor[1], Luca Mottola[2], Cecilia Mascolo[1], Gian Pietro Picco[3], Stephen Ellwood[4], and David Macdonald[4]

[1] Computer Laboratory, University of Cambridge, UK
[2] Swedish Institute of Computer Science, Sweden
[3] Dipartimento di Ingegneria e Scienza dell'Informazione, University of Trento, Italy
[4] Wildlife Conservation Research Unit, University of Oxford, UK

Abstract. We target application domains where the behavior of animals or humans is monitored using wireless sensor network (WSN) devices. The code on these devices is updated frequently, as scientists acquire in-field data and refine their hypotheses. Wireless reprogramming is therefore fundamental to avoid the (expensive) re-collection of the devices. Moreover, the code carried by the monitored individuals often depends on their characteristics, e.g., the behavior or preferred habitat. We propose a *selective* reprogramming approach that simplifies and automates the process of delivering a code update to a target subset of nodes. Target selection is expressed through constraints injected in the WSN, triggering automatic dissemination of code updates whenever verified. Update dissemination relies on a novel protocol exploiting the *social* behavior of the monitored individuals. We evaluate our approach through simulation, using real-world animal and human traces. The results shows that our protocol is able to capture the social network structure in a way comparable to existing offline algorithms with global knowledge while allowing runtime adaptation to community structure changes, and that existing dissemination approaches based on gossip generate up to three times more network overhead than our socially-aware dissemination.

1 Introduction

Wireless sensor networks (WSNs) are increasingly being used to monitor mobile entities in domains ranging from wildlife monitoring [16, 20] to human health-care [22]. In these contexts, WSN nodes are physically attached to animals or people being monitored. Therefore, unlike traditional WSN architectures where all nodes perform a single system-wide task, in these mobile WSNs the code running on a node is often specific to the monitored individual, and may change over time according to the individual's behavior or context. As an example, WSN devices attached to wildlife species (e.g., zebras [16], turtles [13], or badgers [10]) are currently used to study various aspects of their behavior. In the early stages of the deployment, all nodes monitor the same quantities for domain experts to get an initial insight, which can then be used to re-task some of the nodes to further study certain quantities. For instance, the devices carried by badgers that stay close to their burrows may be used to study the environment around the burrows themselves and explain why this subset of animals are following specific

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 178–193, 2010.

paths in the forest instead of others, and how their movements depend on the climate. However, re-capturing the animals to manually re-program the nodes would be very costly, if at all possible.

Techniques for run-time reprogramming of WSNs do exist [33]. However, they fail to tackle two fundamental challenges of the application domain we target:

- The area where monitored individuals dwell is likely to extend beyond the communication range of current sensor devices. Thus, the network is most often characterized by *intermittent connectivity* among the mobile WSN nodes [23]. This prevents re-using well-established solutions for static networks [25].
- The few solutions addressing mobile WSNs disseminate code updates to the entire network, and are therefore ill-suited for a *selective* dissemination of code updates to a target subset. Indeed, the updates would reach more nodes than necessary, wasting resources and reducing lifetime.

On the other hand, animals and humans are *social* beings, with recognizable patterns of movement and community interaction, that can be exploited as a vehicle for delivering code to the intended targets. The core contribution of this paper is a novel approach to selective reprogramming in highly-disconnected, mobile WSNs that, based on the individual's interactions *detects communities at runtime, and exploits their existence and relationships towards efficient update dissemination*. For instance, a single WSN node attached to a badger known to roam often between two communities (i.e., a so-called "central" badger, with a socially-bridging role) can be enough to disseminate code from one community of badgers to the other. In our approach, communities are discerned entirely at run-time. This sets us apart from the few existing dissemination approaches based on social communities, that rely on offline centralized protocols [5] or are otherwise unable to adapt to all changes to the social community structure [14,7,31].

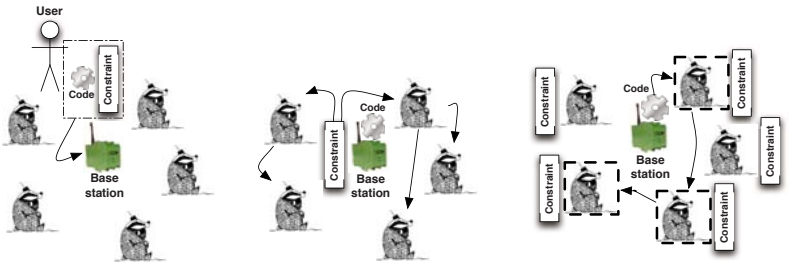
An overview of our approach is provided in Sec. 2, where we introduce a sample scenario showing how a user can target a set of nodes of interest. In Sec. 3, we give details of how the protocol is able *automatically* select these nodes, and deliver the code efficiently.

In Sec. 4, we evaluate the effectiveness of our solution through simulations using animal and human traces collected in real-world experiments. We review related approaches in Sec. 5, and provide brief concluding remarks and directions for future work in Sec. 6.

2 Reference Scenario and System Overview

We illustrate the overview of our approach hand-in-hand with a reference scenario that provides the main application focus for the entire paper. Although the scenario is drawn from the wildlife domain, our techniques are applicable to other mobile WSN scenarios, as we show in Sec. 4 by applying them to human interaction traces. Next, we describe how users specify persistent, network-wide constraints identifying the subset of nodes targeted by reprogramming.

Reference scenario. Fig. 1 depicts the phases of our reprogramming approach in a reference scenario concerned with badger monitoring. As shown in Fig. 1(a), reprogramming entails generating a bundle containing *i)* the code update to be installed on a target



(a) (b) (c)

Fig. 1. Sample scenario showing: (a) code and constraint injection at base station, (b) constraint dissemination to all nodes, and (c) delivery of code to selected nodes (in dashed squares)

subset of the WSN, and *ii)* the *constraint* that identifies these target nodes by means of logical expressions involving their properties. For instance, the constraint may single out only the nodes attached to badgers that spend most of their time close to a cold burrow. The constraints are encoded in periodic beacons for transmission. The bundle is then injected at the base station, or at any other node.

The two constituents of the bundle have a different fate, as show in Fig. 1(b). The constraint is spread to all WSN nodes. Upon reception, a node matches the constraint against its local state, and re-evaluates it periodically. The code update, on the other hand, remains at the base station until at least one node matches the constraint. When this happens, our socially-aware protocol (described in Sec. 3) disseminates the code update only to the target nodes matching the constraint, as shown in Fig. 1(c).

It is important to note that reprogramming can be requested even when no node matching the characteristics specified by the constraint currently exists. In the mobile setting with intermittent connectivity we target, it would be difficult (if not impossible) for users to know and await the moment when the target subset is not empty. Our solution enables users to rely on the system to detect the presence of target nodes *automatically*, by self-adapting to changes in the state of nodes. For instance, one might define constraints to target nodes roaming around different burrows, and inject the code before any node satisfies the constraint. The code will stay at the base station until such behaviour is detected, and will be delivered automatically.

Specifiying constraints. The constraints identifying the target subset are expressed through dedicated constructs. We characterize the state of nodes based on *attributes*. These are name-value pairs describing properties of a node, e.g., the current location or the gender of the individual it is attached to. The construct **`attribute`**`(NAME)` declares an attribute, registered by the run-time layer that takes care of updating the associated value. For instance, in the case of a `LOCATION` attribute, the run-time periodically queries the attached GPS device, and stores the value time-series in memory.

Selecting badgers that stay around cold burrows can be specified as

```
constraint(n_occurrence(LOCATION == burrow) > loc_threshold &&
           avg(TEMPERATURE) < temp_threshold)
```

where `LOCATION` and `TEMPERATURE` are attribute names, and `burrow` is an encoding of a burrow's location in some coordinate system. The built-in functions **`avg`** and **`n_occurrence`** are made available by the underlying run-time support: the latter returns the number of occurrences in an attribute's time series that match the boolean condition given as argument. We provide several built-in functions (e.g., **`avg`**, **`max`** and **`min`**) covering a range of common constraints.

A constraint essentially specifies a boolean function that establishes the membership of a node in a given subset (`constraint(TRUE)` targets the entire WSN). This addressing scheme is well-suited to our scenarios where the target subset changes based on the *state* of nodes—that we capture through attributes, and could hardly be captured through node identifiers. Similar approaches exist in the literature [25, 34]. However, their supporting communication layer targets only static WSNs, while we bring the expressive power of attribute-based node selection into mobile WSNs, as discussed next.

3 Socially-Aware Dissemination of Code Updates

Once the appropriate constraint is stored at the base station, the problem is to efficiently disseminate the code update to the corresponding target nodes. In principle, this could be done using direct transmissions, however in our scenario, we cannot ensure that all the nodes come in range of the base station due to the limited power and radio range of the devices.

In our dissemination protocol, code updates are relayed opportunistically from one animal to the other upon contact. However, unlike existing approaches that propagate updates to the entire network, we limit dissemination as much as possible to the target nodes. This substantially reduces the network overhead, as evaluated quantitatively in Sec. 4. To achieve this goal, we use a characteristic common to many mobile WSN scenarios, namely, the fact that the monitored individuals exhibit *social behavior*. The implicit structure of social interactions, once elicited, provides an effective tool for steering efficient routing decisions. In the rest of this section we describe the aspects of social interaction that are relevant to our goals, along with the way we exploit them in our dissemination protocol. The social foundation of our protocol holds for many animal species [2] [30], including humans [14].

3.1 Overview

Social foundation. A social network is a logical structure of entities tied by some social relation, e.g., friendship. These networks are characterized by strong clustering [7, 17]. Members of a cluster, or *community*, are usually closer to each other socially, than to the rest of the network – i.e., they interact more and spend more time together. Communities tend to be stable over time, although they occasionally vary. An example is animals sharing the same burrow or foraging in the same areas: when cubs grow up, at some point they separate and move to a different area. Moreover, not all members of a community behave the same way, some animals/people are more active or popular than others.

We use the highly mobile and more socially central members to aid the dissemination, since they are more likely to meet other individuals. We call these nodes *leaders*.

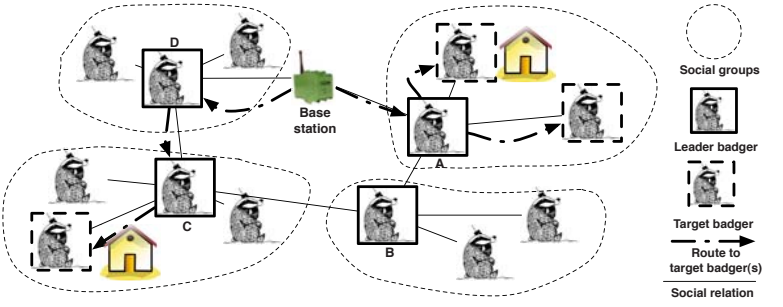


Fig. 2. Example of social communities and their leaders. The target subset includes nodes often visiting a specific area.

Protocol operation. We assume that the base station, where the bundle containing the code update and constraint reside (Sec. 2), is placed in an area where one or more animals dwell. Animals identified as *leaders* are used to carry code updates to communities where at least one member is in the target subset as shown in Fig. 2.

Our protocol *dynamically* identifies communities and leaders in a fully decentralized way, as discussed next. As illustrated in Fig. 2, communities and leaders determine a logical topology where links represent spatio-temporal relations between two individuals, essentially denoting that they are frequently co-located. We exploit these links to disseminate the code updates according to the forwarding rules described in Sec. 3.4.

3.2 Identifying Communities

Social foundation. Members of the same social community are co-located according to a regular pattern and for long periods of time. For instance, at night, badgers roam independently. During the day, however, they tend to congregate around in burrows, where they sleep. Animals using the same burrow tend to spend considerable time together and are therefore often associated to the same community. Our definition of community is a set of nodes spending a certain percentage of their time together.

Protocol operation. To identify communities, we need to quantify the extent of co-location between nodes. To do so, nodes send periodic beacon messages to discover neighbors. Upon receiving a beacon, a node increments by the beacon interval the *contact time* relative to the sending neighbor. This quantity is divided by the time since the first detection of the same node, yielding a *contact ratio* measuring how frequently the two nodes are co-located. Higher ratios indicate more frequent co-location. As time elapses, the contact ratio becomes an accurate indicator of the amount of interaction between two animals. This metric is better at capturing dynamic changes in the community structure than the often-used total-contact duration [14], since it captures not only the order of encounters, but also is able to decay if two nodes become separated.

To create and maintain communities, all nodes send periodic beacons and evaluate each other's contact ratios, which are embedded within these beacons. Two nodes are considered part of the same community when their contact ratios cross a given threshold. For instance, the aforementioned behavior of badgers, sharing the same burrow for

about half of the day, can be modeled by setting a 50% threshold. This indeed corresponds to nodes that are in contact for about half of the time. Thresholds are expected to be defined by domain experts, e.g., based on the species under study. If the ratio crosses the threshold and neither node is yet part of a community, the node with the smaller identifier creates a unique community identifier and includes it in subsequent beacons; the other node joins the new community upon receiving the beacon. If either node is already part of a community, the other joins the same one. If they belong to different communities, the node in the community with fewer members joins the larger one. To enable these decisions, beacons also carry the community size. Our mechanism captures the time evolution of social relations among individuals as nodes can join and leave communities.

An important observation is that the dissemination protocol uses one layer of clustering. More precisely, a node is either a member of a community or not, we do not consider nodes belonging to multiple communities. One can argue that this applies to animals [2], but not for humans. While similar approaches have been adopted for human networks [14,31], human social structures are more complex. If the target application heavily involves membership in multiple communities, our protocol would need to be properly extended to cater for it.

3.3 Identifying Leaders

Social foundation. The behavior of members of the same community may differ [29]. Moreover, this behavior can change over time. For instance, during mating season, adult male badgers travel further from their burrow than other community members, looking for females to mate. Therefore, they are more likely to meet badgers from other communities.

Protocol operation. To accurately and dynamically identify leaders within a community, every node keeps track of two quantities:

- Its *total neighbor count* N, i.e., the number of all distinct nodes it has ever met.
- Its *change-degree of connectivity* C, i.e., the number of neighbors it acquires or looses within a time window.

The two metrics account for different aspects, and leaders should score high in both. For instance, a node with high neighbor count can probably reach many members of its community. The same node, however, may have a low value of change-degree of connectivity, e.g., if it does not move often. This node is not well-suited as a leader. The relative weight of the two metrics must be tuned by domain experts based on the species under study. This is achieved by defining a single *leader score* as $L = \alpha N + (1-\alpha)C$, and properly setting the weight α. In this paper, unless otherwise noted, we use $\alpha = 0.5$. In principle, other metrics could be used, e.g., ego-centrality and betweenness [7]. However, our priority was to disseminate updates as quickly as possible, therefore we focused on identifying the most mobile nodes. Further, an improvement on the neighbour count metric is to use a sliding time window, and consider the neighbour count in this window only. Though we did not use this method in this paper, it is our intention in the future.

Nodes that do not belong to any community or are not associated with a leader (e.g., at start-up or when the community threshold is not reached) are considered leaders of a fictitious community of size one. When a real community with more than one member is created, the node with the highest score L becomes its leader. The identifier of community leader and its score are embedded within beacons, and broadcast to the 1-hop neighbours of the leader, while nodes who are in direct contact with the leader beacon a score $L = 0$. This ensures that each node in a community is logically one hop away from a leader, since the node with the local maximum score is always chosen. If a node in a community finds its score to be higher than that of the current leader, it takes over the leadership. The same processing applies when a node joins a community.

Leaders do not need to be unique in a community. Although an unlikely situation, it may happen that the leader identifier and score are too slow to disseminate for this information to stabilize. Nonetheless, the presence of multiple leaders with similar scores is not problematic in the dissemination process, described next.

3.4 Code Dissemination

The process of disseminating code updates is logically divided in two steps. First, the opportunistic routes leading to nodes in the target set are determined. Then, the actual code is disseminated along these routes. In practice, however, the latter step is pipelined with the former to reduce latency.

Route establishment. The routes are determined by the constraint selecting the target subset. Constraints, encoded in a compact form, are disseminated to all nodes in the network by piggybacking them on beacons. Upon receiving a constraint, a node evaluates whether it belongs to the target subset. If so, it informs its community leader whenever in range.

LeaderID	Target	NextHop	Distance
A	Yes	Base	2
B	No	C	2
C	Yes	C	1
D	Yes	-	-

Fig. 3. Routing table of node D in Fig. 2

Leaders use this information to build routing tables like the one in Fig. 3, based on the network shown in Fig. 2. Besides a leader's own entry, the table is populated by exchanging entries with other leaders whenever they meet, through the periodic beacons. The `Target` field indicates whether at least one member in a leader's community is targeted by the constraint. The `NextHop` field identifies the leader that forwarded a given entry. The `Distance` field is the hop-count measure of how "far" a leader is. Multiple constraints can be disseminated in parallel, distinguished by a unique identifier carried by beacons and used to index multiple routing tables at each node.

Update dissemination.

Update dissemination is governed by the following rules:

- a non-leader can only update its own leader;
- a leader can only update other leaders and the members of its own community.

These rules ensure an efficient dissemination, as shown in Sec. 4, as well as consistent delivery. All leaders (including nodes without a community) receive the update. All other nodes in the target set (i.e., the community members) receive the update from their leader.

Updates follow the routes stored in the leaders' routing tables. Consider for instance Fig. 3. When node D receives an update to be disseminated, it determines through the `Target` field that some of its community members are selected by the constraint, along with members of C's and A's communities. To deliver the update to the selected community members, D waits until it becomes co-located with a sufficient number of community members that require the update (i.e. it receives beacons from these members). These can then receive it simultaneously through broadcast, reducing the communication overhead.

This makes sense for species where the probability of colocation is reasonably high. However, this policy may be revised and the leader could decide to broadcast more often, for example when a given percentage of the required members are present. To reach A and C, D looks at the `NextHop` field in its routing table: the code update is forwarded the next time D meets with C or the base station, respectively.

As constraints are piggybacked on beacons, they propagate faster than code, which is often larger. The routing tables are therefore usually built before the code arrives. If not, the code is buffered until at least one positive value appears in the `Target` field.

Short-lived vs. persistent updates. Constraints and code updates are associated to a version number and a time-to-live (TTL). The version number avoids duplicate delivery. Constraints are re-evaluated periodically and the corresponding entries in the routing table are retained until the TTL expires. When a node matching the constraint is detected, our protocol automatically starts the code dissemination following the mechanisms described. Along with this *short-lived* updates, which disappear from the network after a given time, we also easily support *persistent* updates by setting an infinite TTL. In this case, our scheme caters for a powerful way to make the system self-adapt.

3.5 Implementation Highlights

Our current prototype is based on the Contiki [9] OS, targeting TMote Sky nodes. The system is composed of three core components. A `Communication` component is responsible for building and maintaining routing information. Specifically, it maintains the neighbor table, calculates the contact ratio for every neighbor, and maintains information on the leaders. In addition, the module is also responsible for the reliable delivery of the code updates. To do so, we use a simple broadcast mechanism based on a RTS/CTS mechanism and acknowledgments sent back by the target nodes. A `Constraint Evaluator` module parses received constraints and checks them against the current values of node attributes. This determines whether the local node is included in the target subset. Finally, a `Reprogramming` module dynamically links received code updates (typically of size 2-10 Kb) using the hooks available in Contiki.

4 Evaluation

We first compare the effectiveness of our distributed community detection protocol against a centralized algorithm based on global knowledge of the social graph. The two schemes have similar performance in terms of communities detected, yet our distributed solution is able to detect dynamic changes in the community structure. Next,

we assess how community knowledge improves code dissemination. Based on this, our protocol reduces network traffic by a factor of 66% compared to a gossip protocol.

General settings. We used a one month subset of both the Reality Mining traces [12] and mobility traces from a badger-monitoring deployment [10]. The former include proximity information gathered using 43 mobile phones carried by people moving on a university campus. The latter are collected from the movements of 32 badgers equipped with RFID collars and 28 RFID readers deployed in a forest. These data include time-stamped detection of animals by readers at specific places. Therefore, there would be no explicit information on the connectivity *between* the RFID tags carried by the animals. We convert these traces into connectivity information by considering the nodes within wireless range when the animals are detected by the same RFID reader within a 5-minute time-window. Further, we assume animals stay at the burrow between the time they enter and exit - even though the RFID is unable to detect them underground.

The traces present a different radio model from the traditional WSNs, however here we are more interested in the social model governing the movements of the nodes, rather than modeling the radio, and the these traces are ideal for the former.

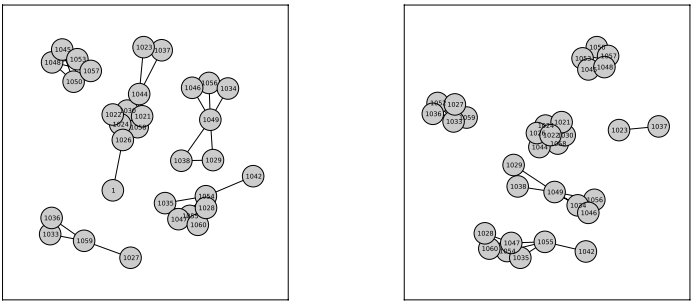
We use the Cooja simulator [27], along with a plug-in we implemented to replace the propagation model in the simulator based on the aforementioned mobility traces. In the community detection protocol, we set the community threshold to 50%. We chose this threshold based on the trace set: badgers sleep during the day in their burrows, therefore they are co-located for at least half a day every day. The threshold is also a good representation of human contacts: if two people spend more than half of their time together, they are more than likely to belong to the same social group. An investigation of the effect of the choice of the threshold is reported later in this section.

4.1 Community Detection

On the badger trace set we compare the performance of our community detection protocol against a well-known algorithm based on modularity optimization [1]. This runs in a centralized fashion and requires global topology knowledge. The communities identified by this algorithm largely reflect the findings obtained through direct observation by the zoologists involved in the study.

Modularity optimization algorithm. Given a specific partitioning of a graph, modularity measures the density of links inside every partition with respect to links between partitions. Higher values correspond to configurations with dense connections inside partitions and sparse connections between different ones. When applied to the study of social networks, partitions are naturally mapped to communities.

The algorithm we consider explores different community configurations to optimize modularity. Initially, every node is in its own community. For every pair of nodes, the algorithm examines the modularity gain obtained by moving either of the two nodes in the other's community. The communities are then changed to maximize this gain. This process repeats for every pair of nodes until no further improvements are achieved. Next, the algorithm creates a new graph with nodes which are the communities found earlier, and the link weights are the sum of the weights of links between the original nodes in the two communities. The algorithm then re-applies the first step on the new graph. The process continues until no further improvements are possible.



(a) Communities after five days. (b) Communities after twenty days.

Fig. 4. Communities found using the contact ratio as metric for link weight

The input to the algorithm is a social graph where there is a link between two nodes if they meet at least once during the simulation time, and the link weights are the ones calculated by our protocol.

Results. We consider different points in time in the badger trace set. Our solution uses the contact ratio to detect dynamic changes in the community structure, therefore, we run the centralized algorithm using this figure as link weight. In this case, both schemes identify the *same* communities after one, five, and twenty days of traces. The communities found after day five are shown on Fig. 4(a). Nevertheless, our distributed solution runs *inside* the network. The centralized algorithm, on the other hand, may run only at the fringes of the system because of significant computational demands. In addition, it would require periodic topology discovery to provide global information as input. This is hardly possible in a mobile scenario with intermittent connectivity.

Even if the conditions to run the centralized algorithm were satisfied, however, the distributed nature of our scheme brings a unique advantage: that of immediately recognizing changes in the community structure. For instance, in the badger scenario the community structure does not change much after day five. This might appear as the long-term behavior. However, by day twenty we see a new community emerging, as shown in Fig. 4(b). Our scheme immediately detects this change, as it is running right on the WSN devices whose behavior caused the formation of an additional community. The centralized approach would identify the new community with significant latency and high overhead, due to the need of periodically collecting global topology information.

4.2 Code Dissemination

We study the performance of our selective code dissemination protocol against state-of-the-art solutions. We compare our approach against:

– the *GCP* [4] gossip protocol for code propagation in mobile sensor networks. This protocol is agnostic of selective dissemination and distributes the update to every

node. To do so, it uses a token-based mechanism to limit the number of transmissions per node, forwarding a code update to any node in range provided the sender still has tokens to spend.
- a constraint-based gossip protocol we implemented. Like ours, this uses the constraints to identify the nodes requiring a code update. The difference with ours is the lack of community knowledge. A node forwards a code update to a nearby device only if *i)* the neighbor belongs to the target subset, or *ii)* the neighbor met a node in the target subset within a specified period (set to half a day). The latter is required to reach nodes in the target subset that may never be in contact with a sender.

Using version numbers, neither protocols transmit a code update if the intended receiver is already equipped with it.

Settings and metrics. A code update consists of a variable number of packets. Each packet is 128 bytes long. We inject the code update at a random node 5 days after start-up. This delay is necessary for the communities to stabilize. We define the target subsets as a given percentage of nodes out of the total. Based on this value, each simulation run considers a different subset to avoid biases due to the subset chosen. GCP is equipped with 15 tokens per node, after we experimentally verified that this value provides a good trade-off between network traffic and overall delivery. For all protocols, we used a one-minute beacon interval for neighbor discovery.

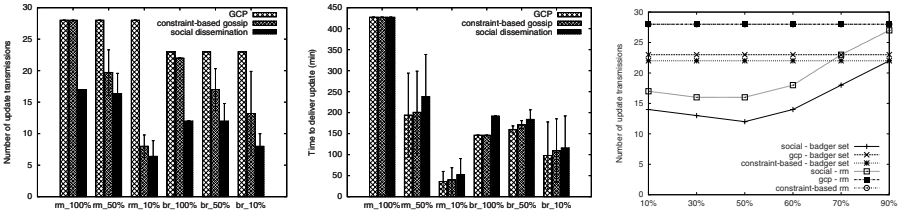
Based on this setting, we measure the following quantities:

- The code update *delivery*, defined as the fraction of nodes in the target subsets that receive the code update. This essentially measures to what extent the dissemination protocol achieves its goal.
- The number of code update *transmissions*, namely the number of bulk data transfers performed during a simulation. This indicates the cost—at the network level—to reach the protocol goal.
- The *latency* required to reach the nodes in the target subset, which provides a complementary measure of cost from a user perspective.

We considered message transmissions as opposed to radio-on-time to evaluate the energy cost of our protocol. Our protocol does not assume that the radio is always on, and is independent of any underlying MAC protocol duty cycling the radio, as long as it provides the ability to discover neighbors and to perform bulk-transfers. There are already efficient MAC protocols for WSN such as [3,11], and it is also easy to see how the social cluster information could be used for duty cycling the nodes - this is however subject of a future work. Further, we do not consider beacons, as all three protocols send them at the same rate. All protocol messages are embedded in beacons, therefore they do not pose additional overhead (the beacons of GCP are, however, 21 bytes lighter).

We run 20 repetitions for each setting. The following results are averages over these repetitions, while the error bars represent the standard deviation around the average.

Results. Hereafter, we show results obtained with code updates of 10 packets. We verified that changing this figure within the range of 5-20 does not influence our results. This is because the bulk transfer of a code image takes little time compared to node mobility, and always completes before the two nodes disconnect.



(a) Overhead (b) Latency (c) Threshold

Fig. 5. Overhead, latency and the effect of clustering threshold of our protocol compared to GCP and constraint-based gossip

In all simulations, the three protocols always deliver the code update to all nodes in the target subset. To do so, however, they incur in drastically different costs at the network level. Figure 5(a) shows the number of code update transmissions against varying target subsets. On average, our community-based protocol improves by a factor of 3.1 and 1.8 over GCP and constraint-based gossip, respectively. However, the gains are smaller as the cardinality of the target subset decreases. This is because the leader nodes that carry code around are a fixed cost that we must pay to reach every part of the system. The impact of this cost is greater as the target subset is smaller. As expected, GCP exhibits the same performance regardless of the target subset. Indeed, it stops only when all nodes are reached, even if the ones in the target subsets already received the code update. constraint-based gossip improves on this behavior, as it may stop earlier if there are no more nodes in the target subset requiring the code update.

To achieve this performance, the community-based protocol trades-off transmissions for latency. The latter is shown in Figure 5(b). Nevertheless, the increased latency in our protocol is limited given the absolute values at stake. On average, we have an increase of a factor of only 1.3 in delay compared to GCP, while the worst case is an increase of a factor of 2.6. GCP shows the best performance in this metric, as it has no restrictions on when to forward a code update. Therefore, it takes advantage of every opportunity, at the cost of redundant transmissions. In our protocol, instead, the leader node knows which nodes in its community need the update, therefore it can wait until it is collocated with these nodes. Once they are all in range, the leader node can update them in one go using broadcast transmissions.

In presence of intermittent connectivity, it may take a long time for some nodes to receive the updates. In the case of targeting 50% of the Reality Mining trace set, this results in a large variation in latency, but some variation is also observed in other cases. This is a characteristic of the network, and affects all three protocols.

We also investigate the behavior of leader nodes, as they play a critical role in our solution. Particularly, we study whether their use may lead to an uneven degradation of available energy among the nodes, e.g., because leaders need to handle more network traffic. To do so, we examine the average number of code updates that leader nodes deliver in our solution, compared to the number of nodes in GCP and constraint-based gossip that deliver an update at least once. We found that the average number of update transmissions a leader sends is 2.3 in the reality mining and 1.2 for the badger trace set.

Both GCP and the constraint-based approaches send 3 and 1.3 update transmissions on average per node, for the reality mining and badger trace set, respectively. We conclude that the leader nodes are *not* depleting their resources more quickly compared to other solutions. Particularly, the leaders we identify largely correspond to nodes that—because of the patterns of colocation—would deliver the code updates anyways. On average, 87% of leader nodes deliver code updates also in GCP and constraint-based gossip. However, our community-detection mechanism identifies *apriori* such nodes. By doing so, we can make them wait for a good opportunity, e.g., when they are in contact with members of their community, to save on unnecessary transmissions.

To further study the effect of leader selection, we also compared our results from targeting the entire network to the performance of the same protocol with random leader selection. This scheme selects leaders randomly from the members of each group. While our overhead is 56% of that of GCP when targeting the entire network, averaged over the two trace set, the random leader selection uses 84%. It is still better than GCP, since the protocol can take advantage of the colocation of the community members, though uses more than necessary transmissions to deliver the code to the communities.

We have also analyzed the effect of the threshold on which communities are separated, which for this analysis has been 50%. In Fig. 5(c) we plot the number of updates sent by all three protocols on both the reality mining and the badger datasets, targeting the entire network. As it can be seen, the threshold choice does affect our results: a different threshold means different community structures and a different number of leaders, thus leads to different overhead. Note, however, that even in the case of bad choices of thresholds, the performance falls back to that of the gossip-based protocols.

5 Related Work

Social routing. A few recent approaches leverage social-inspired metrics for routing. SimBet [7] achieves efficient data dissemination by exploiting "betweenness", a measure of how an individual may socially connect other entities not necessarily known to each other. Bubblerap [15] and Island Hopping [31] use a centralized algorithm to detect communities, based on global knowledge. Bubblerap describes also a distributed extension which detects communities at run-time only if their cardinality grows over time. Thus, every node is bound to the first community it is mapped to, missing the dynamic evolution of social interactions.

In contrast to these approaches, our solution detects communities at run-time and in a fully decentralized fashion. Moreover, we are able to adapt to dynamic changes in the community structure and in the mapping of entities to communities. These features are pivotal to leverage communities for routing in the scenarios we target.

Delay tolerant routing approaches use notions of previous encounters and mobility patterns to decide on best message carriers [19,28,32]. This approach was also extended to mobile sensor networks [28]: while the use of mobility and connectivity to identify good carriers is shared in our approach, with respect to dissemination we go one step further and use community knowledge to improve on the number of messages needed to spread the updates.

WSN reprogramming. To the best of our knowledge, our work is the first to provide a solution for *selective* code dissemination in *mobile* sensor networks. However, the literature includes a wealth of approaches for system-wide reprogramming in static networks [33]. For instance, Trickle [18] disseminates code updates using a "polite gossip" technique to suppress redundant transmissions. The rate of control traffic is adjusted at every device based on the state of neighbor nodes. As neighborhoods keep changing in the scenarios we target, a similar solution would be very inefficient.

Solutions for selective code dissemination in static networks also exist. For instance, Figaro [26] allows selecting subsets of nodes based on node attributes. It employs a tree-based routing scheme for code dissemination, which is difficult to apply in a mobile, disconnected scenario like ours. In TinyCubus [24], code is disseminated to all nodes with a given role, e.g., all cluster-heads. At the network level, TinyCubus assumes a priori knowledge of the system topology, as it requires to specify an upper bound on the number of hops separating nodes with the same role. Such scheme is hardly applicable in presence of dynamic topologies and intermittent connectivity.

In a mobile setting, Impala [21] leverages gossip dissemination to distribute code updates to every device. Version numbers are used to cater for eventual delivery. GCP [4] also targets system-wide reprogramming in mobile sensor networks, using a polite gossip technique similar to Trickle. However, GCP limits network traffic using a token-based scheme whereby nodes can transmit only if they possess enough tokens. ReMo [8] focuses on both static and mobile networks, using physical-layer metrics such as the Link Quality Indicator (LQI) [6] to establish routes for code dissemination. Although these solutions target scenarios similar to ours, they still do not tackle the problem of selective code dissemination. Therefore, being unaware of the selection criteria specified by our users, their use would correspond to significant energy waste.

6 Conclusion

We presented a system for selective reprogramming in mobile WSNs, based on social community detection. Our solution allows users to target a subset of the WSN nodes using constraints on node attributes. A dedicated protocol exploits the social interactions among the monitored entities to disseminate code updates efficiently. We evaluated our framework through real mobility traces. The results showed that, although experiencing a small latency overhead, our protocol saves up to 66% of the transmissions even when reprogramming targets the entire system. These performance gains increase when targeting a subset of the nodes, by virtue of our routing strategy that builds routes to the target nodes based on the social communities. Our future work includes deploying the system on animals in the context of a wildlife monitoring project.

Acknowledgments

The work described in this paper was partially supported by ESF MiNEMA, EPSRC grants EP/E012914 and EP/C544773, the Autonomous Province of Trento under the call for proposals "Major Projects 2006" (project ACube), CONET under EU contract FP7-2007-2-224053 and Swedish Foundation for Strategic Research (SSF).

References

1. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J.STAT.MECH., P10008 (2008)
2. Brown, J.L., Orians, G.H.: Spacing patterns in mobile animals. Annual Review of Ecology and Systematics 1 (1970)
3. Buettner, M., Yee, G.V., Anderson, E., Han, R.: X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In: SenSys 2006: Proc. of the 4th Int. Conf. on Embedded Networked Sensor Systems, pp. 307–320. ACM, New York (2006)
4. Busnel, Y., Bertier, M., Fleury, E., Kermarrec, A.-M.: GCP: Gossip-based Code Propagation for Large-scale Mobile Wireless Sensor Networks. In: Proc. of the Int. Conf. on Autonomic Computing and Communication Systems (2007)
5. Chan, S.-Y., Hui, P., Xu, K.: Community Detection of Time-Varying Mobile Social Networks. In: Proc. of the First Int. Conf. on Complex Sciences: Theory and Applications, Complex 2009 (2009)
6. Chipcon Tech. CC2420 Datasheet, focus.ti.com/docs/prod/folders/print/cc2420.html
7. Daly, E.M., Haahr, M.: Social Network Analysis for Routing in Disconnected Delay-tolerant MANETs. In: Proc. of the Int. Symp. on Mobile Ad-Hoc Networking and Computing, MobiHoc (2007)
8. De, P., Liu, Y., Das, S.K.: ReMo: An Energy Efficient Reprogramming Protocol for Mobile Sensor Networks. In: Proc. of the Int. Conf. on Pervasive Computing and Communications, PERCOM (2008)
9. Dunkels, A., Grönvall, B., Voigt, T.: Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. In: Proc. of 1^{st} Wkshp. on Embedded Networked Sensors (2004)
10. Dyo, V., Ellwood, S.A., Macdonald, D.W., Markham, A., Mascolo, C., Pasztor, B., Trigoni, N., Wohlers, R.: Poster Abstract: Wildlife and Environmental Monitoring using RFID and WSN Technology. In: Proc. of the Int. Conf. on Embedded Networked Sensor Systems, SenSys (2009)
11. Dyo, V., Mascolo, C.: Efficient node discovery in mobile wireless sensor networks. In: Nikoletseas, S.E., Chlebus, B.S., Johnson, D.B., Krishnamachari, B. (eds.) DCOSS 2008. LNCS, vol. 5067, pp. 478–485. Springer, Heidelberg (2008)
12. Eagle, N., Pentland, A.S.: Reality mining: sensing complex social systems. Personal Ubiquitous Comput. 10(4) (2006)
13. Gorlick, A.: Turtles to test wireless network (July 2007)
14. Hui, P., Crowcroft, J., Yoneki, E.: Bubble rap: Social-based Forwarding in Delay Tolerant Networks. In: Proc. of the Int. Symp. on Mobile Ad-Hoc Networking and Computing, MobiHoc (2008)
15. Hui, P., Yoneki, E., Chan, S.Y., Crowcroft, J.: Distributed community detection in delay tolerant networks. In: Proc. of Int. Wrkshp. on Mobility in the Evolving Internet Architecture, MobiArch (2007)
16. 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: Proc. of the Int. Conf. on Architectural Support for Programming Languages and Operating Systems, ASPLOS-X (2002)
17. Krause, J., Croft, D., James, R.: Social Network Theory in the Behavioural Sciences: Potential Applications. Behavioral Ecology and Sociobiology 62(1) (2007)
18. Levis, P., Patel, N., Culler, D., Shenker, S.: Trickle: a Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. In: Proc. of the Symp. on Networked Systems Design and Implementation, NSDI (2004)

19. Lindgren, A., Doria, A., Schelén, O.: Probabilistic Routing in Intermittently Connected Networks. In: Dini, P., Lorenz, P., de Souza, J.N. (eds.) SAPIR 2004. LNCS, vol. 3126, pp. 239–254. Springer, Heidelberg (2004)
20. Lindgren, A., Mascolo, C., Lonegan, M., McConnell, B.: Seal2Seal: A Delay-Tolerant Protocol for Contact Logging in Wildlife Monitoring Sensor Networks. In: Proc. of Int. Conf. on Mobile Ad-hoc and Sensor Systems, MASS (2008)
21. Liu, T., Martonosi, M.: Impala: A middleware system for managing autonomic, parallel sensor systems. In: Proc. of the SIGPLAN Symposium on Principles and Practice of Parallel Programming (2003)
22. Lorincz, K., Chen, B.-R., Werner Challen, G., Roy Chowdhury, A., Patel, S., Bonato, P., Welsh, M.: Mercury: A Wearable Sensor Network Platform for High-fidelity motion Analysis. In: Proc. of the Int. Conf. on Embedded Networked Sensor Systems, SenSys (2009)
23. Lukac, M., Girod, L., Estrin, D.: Disruption Tolerant Shell. In: Proc. of the SIGCOMM Wkshp. on Challenged Networks, CHANTS (2006)
24. Marrón, P.J., Lachenmann, A., Minder, D., Hahner, J., Sauter, R., Rothermel, K.: TinyCubus: a flexible and adaptive framework sensor networks. In: Proc. of the European Wkshp. on Wireless Sensor Networks, EWSN (2005)
25. Mottola, L., Picco, G.P.: Logical Neighborhoods: A Programming Abstraction for Wireless Sensor Networks. In: Gibbons, P.B., Abdelzaher, T., Aspnes, J., Rao, R. (eds.) DCOSS 2006. LNCS, vol. 4026, pp. 150–168. Springer, Heidelberg (2006)
26. Mottola, L., Picco, G.P., Amjad, A.: Fine-Grained Software Reconfiguration in Wireless Sensor Networks. In: Verdone, R. (ed.) EWSN 2008. LNCS, vol. 4913, pp. 286–304. Springer, Heidelberg (2008)
27. Österlind, F., Dunkels, A., Eriksson, J., Finne, N., Voigt, T.: Cross-level Simulation in COOJA. In: IEE SenseApp 2006 (2006)
28. Pasztor, B., Musolesi, M., Mascolo, C.: Opportunistic mobile sensor data collection with scar. In: Proc. of the 4th IEEE Int. Conf. on Mobile Ad-hoc and Sensor Systems (MASS 2007), Pisa, Italy, October 2007. IEEE Press, Pisa (2007)
29. Ramos-Fernández, G., Mateos, J., Miramontes, O., Cocho, G., Larralde, H., Ayala-Orozco, B.: Lévy Walk Patterns in the Foraging Movements of Spider Monkeys (Ateles geoffroyi). Behavioral Ecology and Sociobiology 55(3) (2004)
30. Sanderson, G.C.: The Study of Mammal Movements: A Review. The Journal of Wildlife Management 30(1) (1966)
31. Sarafijanovic-Djukic, N., Pidrkowski, M., Grossglauser, M.: Island Hopping: Efficient Mobility-Assisted Forwarding in Partitioned Networks. In: Proc. of the Int. Conf. on Sensor and Ad Hoc Communications and Networks, SECON (2006)
32. Small, T., Haas, Z.J.: The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way). In: Proc. of the 4th ACM Int. Symp. on Mobile ad hoc networking & computing (MobiHoc), pp. 233–244. ACM, New York (2003)
33. Wang, Q., Zhu, Y., Cheng, L.: Reprogramming wireless sensor networks: challenges and approaches. IEEE Network 20(3) (2006)
34. Welsh, M., Mainland, G.: Programming Sensor Networks Using Abstract Regions. In: Proc. of the Symp. on Networked Systems Design and Implementation, NSDI (2004)

Improving Sensornet Performance by Separating System Configuration from System Logic

Niclas Finne, Joakim Eriksson, Nicolas Tsiftes, Adam Dunkels, and Thiemo Voigt

Swedish Institute of Computer Science
{nfi,joakime,nvt,adam,thiemo}@sics.se

Abstract. Many sensor network protocols are self-configuring, but independent self-configuration at different layers often results in suboptimal performance. We present Chi, a full-system configuration architecture that separates system logic from system configuration. Drawing from concepts in artificial intelligence, Chi allows full-system configuration that meets both changing application demands and changing environmental conditions. We show that configuration policies using Chi can improve throughput and energy efficiency without adding dependencies between layers. Our results show that sensornet systems can use Chi to adapt to changing conditions at all layers of the system, thus meeting the requirements of heterogeneous and continuously changing system conditions.

1 Introduction

The sensornet community is moving toward modular architectures that allow a clean separation of concerns [3,5,7,9,20]. So far, however, the performance of such modularized designs has been dissatisfying due to problems with cross-layer interactions. For example, Kim et al. write [12]: *"[...] there is still a large performance gap to the raw radio bandwidth that would require a cross-layer design and integration with the MAC and the packet processing in the OS."* Similarly, experience from sensornet deployments [17] has shown the need for configuration across modules and for gathering system statistics.

Cross-layer optimizations have primarily been implemented by coupling the programming interfaces of different components. Hence, using cross-layer designs typically requires that we sacrifice system modularity to improve system performance. As a remedy to this problem, researchers have proposed specialized architectures for cross-layer optimization [13,16,10]. These specialized architectures enable applications to be configured to meet energy efficiency goals. We argue, however, that since energy efficiency is not the only objective of a sensornet, a cross-layer optimization architecture should be able to focus on other metrics as well.

We present Chi, a lightweight architecture that enables cross-layer optimizations in sensornet systems without requiring an unmodular cross-layer design. The main design principle of Chi is that components must not be required to

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 194–209, 2010.

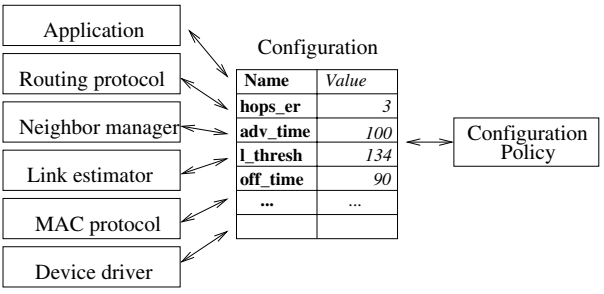


Fig. 1. Chi keeps the configuration data of different system layers in a central component. The configuration can be changed by a configuration policy. Unlike existing cross-layer architectures, individual protocol layers are unaware of the configuration of the other layers.

have any knowledge of parameters exported by other components. Instead, all such knowledge is in separate components that enforce configuration policies. By using Chi, we maintain the separation of concerns between layers while providing the same performance as integrated cross-layer optimizations. In contrast to previous work, Chi takes a generalized approach to configuration that enables systems to be optimized not only to meet energy objectives but also to meet other objectives such as latency, throughput, and sensor coverage.

The Contiki operating system separates protocol logic from protocol headers to achieve network protocol modularity with retained execution-time efficiency [5]. Similarly, Chi provides system configuration modularity with low run-time overhead. We draw from the work made within the autonomic computing community on blackboard systems [4]. The central component in Chi is a blackboard that holds the system configuration and relevant parts of the system state. Storing the configuration in one component simplifies updates made by external configuration modules, as illustrated in Figure 1. In addition to the blackboard component, Chi accommodates configuration policies written to optimize a sensornet toward different objectives.

Our contribution is to show that Chi solves the problem of cross-layer optimization for sensornet systems using a generalized programming abstraction. We demonstrate the abilities of Chi in three case studies, showing that an application using holistic configuration outperforms the same application when using constant parameter settings of the protocol layers, as well as when using several adaptive layers. We show that by using Chi the TCP performance increases by an order of magnitude without putting any cross-layer logic within the networking layers. Instead, the cross-layer optimizations are put into separate configuration policies and are decoupled from all protocol implementations.

The rest of this paper is structured as follows. We present the background of cross-layer design, holistic configuration and blackboard systems in Section 2. We describe Chi in Section 3 and its implementation in the Contiki operating system in Section 4. We evaluate the architecture in Section 5, present related work in Section 6, and conclude the paper in Section 7.

2 Background

Layering separates different concerns in a network architecture and reduces the design complexity. The plethora of routing, transport, and medium access control protocols for sensor networks has made layering the prevalent design choice also in sensor network architectures. The high modularity of layering, however, restricts the collaboration of different layers that could potentially benefit from sharing each others unique information.

2.1 Performance Improvements through Cross-Layer Design

Achieving near-optimal throughput and energy consumption is difficult when using independently designed protocols in different layers. This is easier in vertically integrated systems that benefit from a coordinated design of several layers. Koala [19] and Dozer [2] are examples of vertically integrated systems with duty cycles less than 1% in low power data gathering applications. Their specific target in design, however, restricts them from achieving the same efficiency when adapting to different network traffic patterns.

Cross-layer approaches make existing protocols more adaptive to different workload patterns by allowing the layers to interact and share information. Previous research on cross-layer design for sensor networks has led to improvements in energy-efficiency [8,18]. The negative consequence is that cross-layer design, by definition, increases the coupling between modules.

2.2 Cross-Layer Design Breaks Layering

Cross-layer designs have been criticized as leading to "spaghetti design" [11]. Without layering it is difficult to achieve adequate separation of concerns which may lead to stability problems and negative impact on performance. Additionally, tight coupling decreases the modularity of the architecture and makes it complex to replace modules.

2.3 Holistic Configuration

By holistic configuration, we mean that parameters in all parts of the system can be configured through a separate configuration component, called a configuration policy in Chi. While layered systems usually store configuration parameters in module variables, a separate configuration policy enables simultaneous configuration of the whole system. This makes it possible to add external algorithms that optimize the system using information from multiple layers. The optimization objectives can for instance be low energy consumption, minimum delay, or maximum throughput. Changing the applications objectives is simple because only the configuration policy needs to be updated.

A configuration policy can be implemented in plain C using basic if-then-else statements that optimize the configuration toward the application objectives. It can also consist of a rule engine and a set of rules that are triggered when the system state changes. If the application objectives change, the configuration policy can be replaced to reflect the new objectives.

2.4 Blackboard Systems

A blackboard [4] is a concept used within the artificial intelligence community. Conceptually, a blackboard is a tool used by a group of experts to solve a complex problem. In a software system, the blackboard is a component that typically stores key-value pairs, and has a mechanism for notifying interested components when a value changes. The classic blackboard design consists of a blackboard, a set of independent knowledge sources, and a number of control components. The knowledge sources have both the knowledge and the algorithms needed to solve a specific problem, whereas the control components steer the execution order and triggering. The traits of the blackboard makes it a suitable solution for the requirements of a holistic configuration architecture.

3 Chi: A Full-System Configuration Architecture

We have designed Chi[1], a full-system configuration architecture that uses a blackboard to enable cross-layer information sharing despite keeping modules decoupled. The blackboard provides a shared variable abstraction that is accessed in an independent module in each sensor node. Beside providing a programming interface for accessing variables, the blackboard has a notification process for subscribers of value modifications. As illustrated in Figure 1, modules export their configuration parameters through the blackboard. Configuration policies can then use any of the available parameters to optimize the system for any objectives determined by the application.

Chi is designed to be a dynamic configuration architecture. In a modular system such as Contiki, in which software modules can be loaded at runtime, it is necessary that also the configuration architecture can accommodate new parameters and configuration policies. New insights on protocol optimizations are easily integrated into deployed networks because the configuration policies are replaceable.

3.1 Separating Configuration from Logic

Chi separates system logic from system configuration to make it possible to alter the configuration without having to change the logic. Moreover, system modules do not need to contain any logic for updating their configuration: this service is provided by Chi and the configuration policies used in the system.

Existing mechanisms conflate logic and configuration by storing configuration parameters as module variables. To change the configuration of a module, the internal variables need to be changed. Thus the module must contain logic for storing and retrieving configuration parameters. Without a consistent interface for storage and retrieval of configuration parameters, every module provides its own mechanism for doing so, leading to systems that are difficult to reconfigure.

[1] The name Chi comes from the Greek letter χ, representing the cross-layer information sharing that the architecture enables.

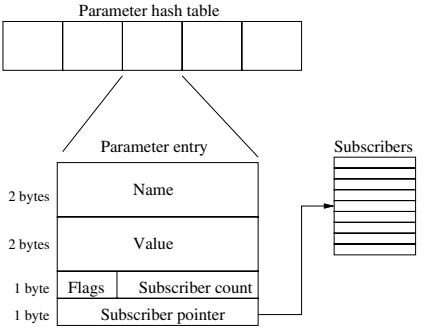


Fig. 2. The memory layout in the blackboard. Parameters are stored in a hash table for fast lookup. Subscribers are listed as function pointers in a separate table.

Chi provides a consistent interface for storing and retrieving configuration parameters from the blackboard, thus making it possible to reconfigure the entire system using a single interface.

Configuration and data sharing abstractions that are aware of the details in specific protocols can be implemented by using the generalized Chi architecture. For instance, a network-based data sharing abstraction such as Hood [22] could store reflected data variables locally in Chi. The need for creating parameters dynamically–which is possible in Chi–is highlighted in the case of deploying a heterogeneous sensor network where different types of sensor nodes may want to share different parameters.

Sensornet protocols are in general unaware of which specific configuration parameters that protocols at other layers make available. A configuration policy, as previously depicted in Figure 1, therefore handles the protocol-specific optimization. If the protocols change, for example by code dissemination and dynamic loading in a deployed system, a corresponding update must be made with the configuration policy. In the evaluation (Section 5), we will show three configuration policies in practice.

3.2 Inter-layer Information Hiding

To achieve a meaningful separation of concerns, it is important that different layers of the system are independent of each other. If modules in adjacent layers would depend on having information about each other, it would be difficult to replace them. Instead of depending on inter-layer information sharing to enable cross-layer optimization, Chi moves the information sharing from the inter-layer interface into the blackboard to keep the layers separate. This ensures that inter-layer interfaces focus on the abstractions provided by each layer and not on information sharing between layers.

3.3 State Monitoring

State information can serve as input for configuration policies. Components monitor different parts of the node state and publish the state information in the blackboard. State can be collected either in the nodes through active monitoring, or it may require communication with neighboring nodes to gain a complete picture of the surroundings of a node.

In some cases, the information published by one component may not be directly usable by others. Chi allows reusable components to be plugged in to process data and to produce a refined output. For example, a network statistics component can take the raw packet statistics produced by the network stack and calculate whether the node is in bulk traffic mode or in passive mode. This information can then be used by a configuration policy to optimize the system based on a refined input.

The parameter subscription mechanism in Chi ensures that interested parties are notified if a parameter value changes. Just holding the shared state would have required that modules periodically poll the blackboard for changes, which would add latency to the information exchange and increase the processing energy. In particular, configuration policies regularly require that components must react within a limited time after the event occurs.

4 Implementation

We have implemented the Chi architecture in Contiki, but the architecture is general enough to be portable to other systems. Figure 2 illustrates the memory layout. The blackboard component uses two tables: the parameter hash table and the subscriber table. Parameters are represented by a name pointer, a value, a set of flags, the number of subscribers, and a pointer to the first subscriber in the subscriber table. We restrict the values to be of integer type to have a concise API. Moreover, we have not identified any need for other types. When compiled for 16-bit computing architectures such as the MSP430, each parameter requires six bytes, whereas the subscriber table requires two bytes per subscriber to store pointers to callback functions.

Chi's API consists of eight functions. The configuration parameters are denoted by textual names, such as "mac.off_time" and "measure.period". The *set* function assigns a value to a configuration parameter. The *get* function obtains the last set value. Whether or not a value has been set is checked with the *exists* function. The *subscribe* function registers a callback function as a subscriber to a specified configuration parameter. The callback function is called whenever the parameter is changed using the *set* function. The *unsubscribe* function removes a previously registered callback.

Parameter values can be read and written without subsequent parameter lookups in the hash table by holding a one-byte index for the parameter. The *lookup* function returns an index value if the parameter exists. Thereafter, the *entry_get* and *entry_set* functions will provide significantly faster access to the parameter by using the index value.

5 Evaluation

We evaluate Chi through a series of experiments to determine whether it achieves the same performance improvements as those of typical cross-layer designs, and whether the dynamic properties of the architecture results in any performance penalty. The first experiment is inspired by a condition monitoring application that we evaluate by comparing the performance of different optimization principles. In the second experiment we show that the separation of configuration and logic makes it possible to reuse optimizations in configuration policies between different applications and communication stacks. The third experiment demonstrates that Chi improves the communication performance as much as that of a specialized architecture for application feedback to the MAC layer. Lastly, we evaluate Chi through a set of micro benchmarks where we measure the cycle count of each operation.

5.1 Case Study: Condition Monitoring with Bulk Transfer

To quantify the effectiveness of Chi, we implement a condition monitoring application in Contiki using a holistic configuration policy. We compare the performance with three other types of network stack designs: constant configuration, adaptive layers, and cross-layer optimizations. The application has the same communication behavior as applications for condition monitoring of industrial motors: it samples large chunks of vibration data periodically, and sends the data to a sink for processing and analysis. A data chunk is typically a few kilobytes large. We apply the configuration policy on the X-MAC protocol [1] and the Rime communication stack [5]. To support bulk transfers over a multi-hop network, we implement a bulk transport layer using Rime's data collection abstraction. We use three Tmote Sky nodes forming a two-hop network.

Constant Configuration: The version with constant parameters uses 20 ms wake time and 480 ms sleep time, resulting in a duty cycle of 4%. On the layers above the MAC layer, we set the routing advertisement interval to 60 s, the data packet transmission interval to 1 s, and the retransmission timeout to 1 s.

Adaptive Layers: Adaptive layers means that each layer optimizes itself using internal knowledge. In this experiment, we implement the version of the X-MAC protocol [1] that adapts to the traffic load using only information about the packets sent and received.

The adaptive layers setup is similar to that of the constant configuration setup, but to avoid collisions, the routing advertisement rate is adapted by delaying the next advertisement by 10 s after receiving a packet. The transport layer adapts the send rate in order to increase the throughput. If there is a large delay between a sent packet and its corresponding acknowledgement, we reduce the send rate. If the delay is below a certain threshold, we increase the send rate as much as possible while keeping the delay below the threshold. A large delay indicates retransmissions in the lower layers and that the next node in the route may be overloaded.

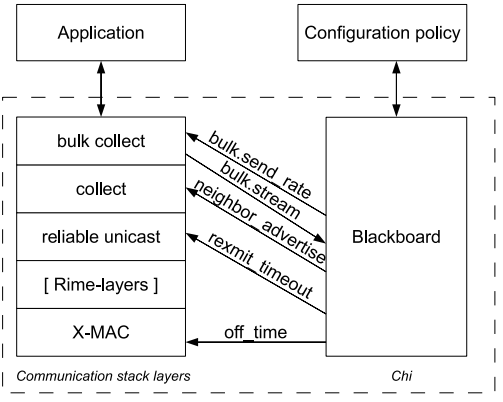


Fig. 3. The condition monitoring application and its parts

A Cross-Layer Design: The cross-layer design in this experiment combines information from the transport layer and the link layer. The consequence is that the bulk transport module must be coupled directly with the MAC protocol implementation. The bulk transport module must know of MAC-specific variables or functions that will no longer be valid if the MAC protocol would be switched. Moreover, the data collection module is coupled directly with the bulk transport module to know when not to send routing advertisements.

In this experiment, the transport layer sets a flag as a global variable and reconfigures the MAC layer at the beginning and end of a bulk transfer. The send rate is fixed at 20 packets per second. While a bulk transport is active, the data collection module withholds routing advertisements, and the reliable unicast layer uses a shorter retransmission timeout of 0.5 s.

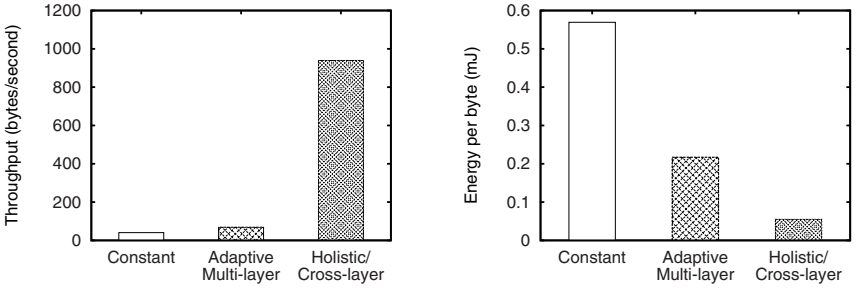
The Chi Design: The Chi design consists of a policy that optimizes the system through a set of parameters in multiple layers. Whereas the cross-layer design described in the previous section modifies the communication stack to fit application requirements, the Chi design uses the communication stack without changing any interfaces or intra-layer logic.

Figure 3 shows the configuration policy that manages the reconfiguration, and Figure 4 depicts the corresponding code. Reconfiguration decisions are primarily affected by the bulk transfer parameter. When the application switches between regular transfer and bulk transfer, it causes a global reconfiguration of the communication stack. This cannot be done with a constant configuration since it would be optimized for either energy efficiency or throughput. Our approach is to use configuration policies optimized for the specific application and reconfigure the communication stack when needed.

Throughput and energy consumption: We show the results in Figure 5. Since the results achieved with the cross-layer design and the Chi design are

```
/* This function is called when "bulk.stream" changes */
void stream_changed(const char *name, int value) {
  if(value != 0) {
    set("mac.off-time", 0);
    set("bulk.send-rate", 20);
    set("collect.routing-advertisements", 0);
    set("runicast.rexmit-time", CLOCK_SECOND / 2);
  } else {
    /* Restore default configuration */
    set("mac.off-time", MAC_DEFAULT_OFF_TIME);
    set("bulk.send-rate", 1);
    set("collect.routing-advertisements", 1);
    set("runicast.rexmit-time", DEFAULT_REXMIT_TIME);
  }
}
/* Register a callback for "bulk.stream" */
subscribe("bulk.stream", stream_changed);
```

Fig. 4. The configuration policy reconfigures the communication stack based on routing information from the network layer. If the system indicates that a bulk transfer is about to occur, the policy sets the system in high throughput mode.



(a) Throughput (b) Energy

Fig. 5. The holistic configuration outperforms the constant configuration and the multi-layer self-adaptive configuration both in throughput and energy

identical, they are shown as one result in the table. When using constant parameters, the throughput is quite low and the power consumption is moderate. The result depends on the chosen parameters that, as discussed above, lead to a duty cycle of around 4%. Although a higher duty cycle leads to higher throughput, the power consumption increases. The adaptive layers design is considerably more efficient than the design with constant parameters. The power consumption decreases to about one third compared with constant parameter setting.

Both the Chi design and the cross-layer design yield a high throughput and a low power consumption. The reason is that the nodes can immediately switch from low power mode to high throughput mode when a bulk transfer begins since the application demands are known by the configuration policy. In the cross-layer design, the same demands are hard-wired into the different layers.

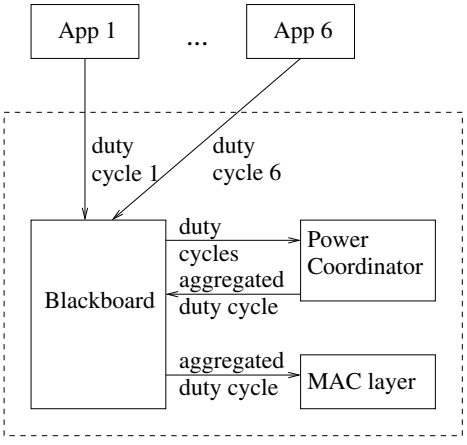


Fig. 6. The power coordinator uses Chi to coordinate duty cycles

5.2 Case Study: TCP Optimization over a Power-Saving MAC Protocol

The second case study shows that we can reuse optimizations of a Chi configuration policy in a different type of application using another communication stack. The application studied is a web-service application running HTTP over TCP/IPv6 [7] with X-MAC as the MAC protocol. The scenario consists of a client that connects to a server, transfers some data, and then closes the connection. For this purpose we use two Tmote Sky nodes that are able to communicate directly with each other.

This scenario differs from the previous scenario since this is a request-response scenario and the first was a bulk transfer. There are also similarities, however, since both scenarios transfer many packets and use X-MAC. By re-using the MAC optimization of the configuration policy from the first case, we can get a much higher performance for the web-service request. The configuration policy is illustrated in Figure 7. The TCP layer publishes the parameter "tcp.connection.count" and the value of this parameter is determined by counting the open TCP connections in the IP stack. The TCP layer needs no knowledge of the MAC layer and vice versa when using Chi–all cross-layer logic is put into the configuration policy.

Figure 8 shows the result of running the experiment with and without configuration policy optimization. As expected, the overhead of setting up a TCP connection decreases in relation to the payload size when more data is transmitted. The data rate increases with an order of magnitude when using the configuration policy. The low performance when using no optimization is caused by TCP waiting for acknowledgments for each sent packet, and both TCP segments and acknowledgments are delayed depending on where in the X-MAC duty cycle each node is. The optimization yields such a high performance by using a 100% duty cycle in X-MAC when there are active TCP connections.

```
/* This function is called when "tcp.connection.count" changes */
void connections_changed(const char *name, int value) {
  if(value > 0)
    set("mac.off-time", 0);
  else
    set("mac.off-time", MAC_DEFAULT_OFF_TIME);
}
/* Register a callback for "tcp.connection.count" */
subscribe("tcp.connection.count", connections_changed);
```

Fig. 7. The configuration policy reconfigures the MAC layer for high throughput when at least one TCP connection is active

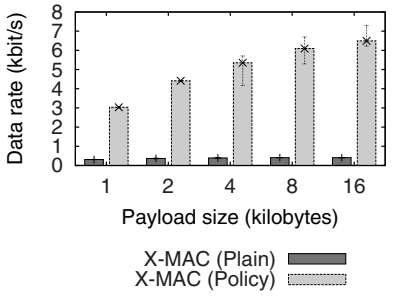


Fig. 8. The data transfer rate of TCP/IPv6 over X-MAC with and without configuration policy optimization

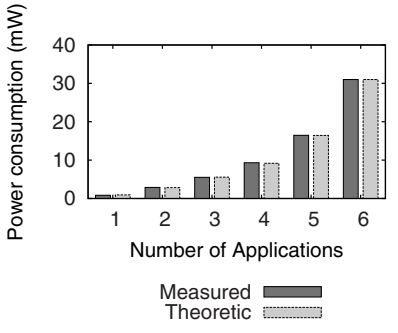


Fig. 9. The measured power consumption of the aggregated duty cycles of multiple applications matches the theoretical values

5.3 Case Study: Aggregation of Multiple Duty Cycles

Data from different sensors can require different communication patterns depending on the deployment. Klues et al. have presented the Unified Power Management Architecture (UPMA) [13], which separates power management from MAC level functionality. UPMA is able to coordinate the duty cycles of multiple applications. Applications store their duty cycles in a Power Management Table, and UPMA uses a configured policy to coordinate the duty cycles.

In this experiment, we emulate the behavior of the aforementioned Power Management Table by using Chi. The experimental setup consists of multiple applications that periodically transmit data according to a configured duty cycle. Applications insert their duty cycles into Chi, as shown in Figure 6. The power coordinator subscribes to the duty cycle parameters in Chi. When an application submits its duty cycle, the power coordinator computes the aggregate duty cycle and assigns this value to the duty cycle parameter in Chi used by the MAC protocol. We use the same duty cycles as Klues et al. in our experiment. The radio on-time is 200 ms for all applications, and the radio off-times are 12.6 s, 6 s, 3 s, 1.4 s, 600 ms, and 200 ms.

In contrast with Klues et al., we do not measure the duty cycle, which is an indirect metric for power consumption, but instead we directly measure the radio power consumption using Contiki's software-based on-line energy estimation method [6]. Using this method, we measured the power consumption of the CC2420 radio as 59.92 mW. The theoretical radio on-time for the six applications with the duty cycles as defined by Klues et al. is slightly below 52% (31.16 mW), matching our measured value of approximately 31 mW.

The results in Figure 9 are similar to those of Klues et al. (Figure 10, [13]) in that the theoretical values match the measured ones. The results confirm that our generalized configuration architecture achieves the same optimized radio power management as that of a specialized architecture such as UPMA.

5.4 Operations Benchmark

We execute a benchmark that measures the required time for the blackboard operations in Chi. We use the internal clock of the MSP430F1611 processor in a Tmote Sky node to count clock cycles. The blackboard is set up with 20 parameters and uses a hash table size with 32 entries. Despite increasing the risk of collisions, the hash table uses a size of 2^n instead of a prime number. Our experiments have shown that the performance degradation of these extra collisions is less severe than the degradation caused by the expensive modulo operation that we avoid this way.

Figure 10 shows the numbers of clock cycles used by the main blackboard operations. The *set* and *get* operations have a simplified interface, but require a parameter lookup in the hash table at each call. In the rare cases where parameters must be accessed several hundred times per second, the *entry_get* and *entry_set* functions provide a shorter path to the parameter by holding a pointer to it in the table. Thus the need to do a lookup is eliminated and the performance becomes comparable to that of a pre-compiled configuration approach such as TinyXXL [15].

5.5 Network Power Consumption

While the micro benchmark gives a clear view of the cost of configuration operations in terms of clock cycles, it is the effect on power consumption in a typical sensor network application that is the key issue. To quantify the power consumption in a sensor network, we conduct two experiments with a data-collection application. We compare the pre-compiled, constant configuration setting with the use of Chi. We measure the power using an online energy-estimation method [6], and collect the energy data when the experiment has finished. The sensor nodes communicate using Rime [5]. When running the experiment with Chi, we substitute calls to Chi for the constant configuration variables in Rime. In addition, we store communication statistics in Chi instead of in memory variables. The experiments are conducted with both a TDMA-based protocol that is specialized for data collection, and a data collection protocol using the more generic X-MAC protocol underneath.

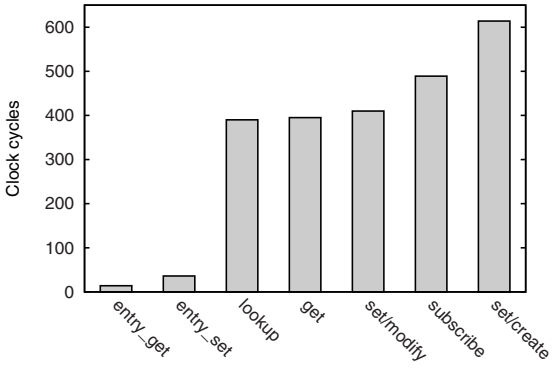


Fig. 10. A micro benchmark for the Chi operations. The *entry_get* and *entry_set* operations are the most commonly used and are therefore optimized for execution-time efficiency.

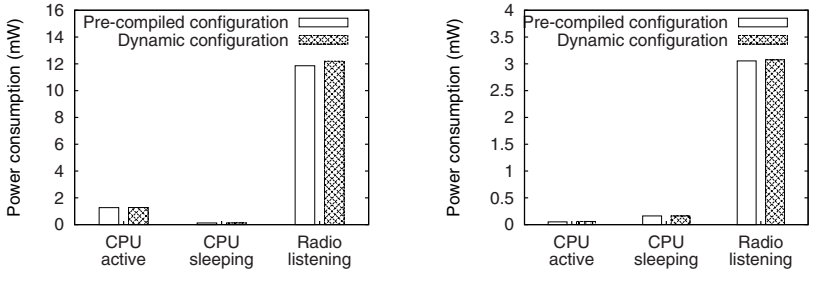


Fig. 11. The dynamic design of Chi has a negligible impact on power consumption. The power consumption is measured for a data collection network using either X-MAC or TDMA. Note that the vertical scales are different.

An X-MAC-based data collection protocol: The application in this experiment measures temperature, humidity, and light, and sends this data to a base station every other second. The system uses the X-MAC protocol [1] as an energy saving MAC layer, and the data collection module in Rime to deliver the data to the sink. Route advertisements are sent once per minute. We measure the average consumed energy at each node in an indoor testbed of 15 Tmote Sky nodes during a period of 50 sensor measurements.

Figure 11(a) shows the average measured power consumption over five test runs with pre-compiled configuration and five corresponding runs with a configuration policy using Chi. The power consumption overhead of the dynamic configuration is on average 2.5%.

A TDMA-based data collection protocol: We build a small data collection network using a TDMA-based data collection protocol named CoReDac [21]. The leaf nodes in our network transmit a packet every 5 s. We measure the power consumption of a node in the middle of the tree with and without Chi. The result in Figure 11(b) shows a negligible overhead of 0.9% for the Chi-based system. As expected, slightly more CPU power is required to handle the variables stored in Chi.

6 Related Work

Although many sensornet protocols are adaptive, they mainly adapt by using intra-layer information. Examples include the scheduled channel polling MAC protocol [24] that changes its duty cycle using on the current traffic load and the MintRoute protocol [23] that changes its forwarding tables based on communication conditions. Independent self-adaptation at multiple layers can lead to sub-optimization where self-adaptation mechanisms at different layers counteract each other [11].

The multitude of non-standard, cross-layer designs have led to various efforts to generalize cross-layer interactions into configuration architectures. Lachenmann et al. present TinyXXL, a language and framework that supports cross-layer interactions [15]. The framework is similar to our work in that it provides a repository for storing system state and configuration. Like Chi, it supports cross-layer interaction and reconfiguration using a publish and subscribe mechanism. TinyXXL is a language extension of nesC, however, and requires recompilation when adding or removing parameters. Köpke et al. suggest using a blackboard for component-based interactions [14], but do not quantify the effects of using the blackboard. We use a similar technique for parameter storage, but focus on policy-based, cross-layer optimizations that retain the tiered networking design used in the Internet and in the Rime networking stack [5].

Several sensor network communication architectures provide mechanisms for inter-layer information sharing in the communication stack. SP [20] allows information to be shared between the link layer and the network layer. The modular network architecture by Cheng et al. [3] is a decomposition of sensornet protocols into common modules that can be shared at multiple layers. Chameleon [5] uses packet attributes to provide packet-based information sharing across layers while maintaining the separation of concerns as traditional layered architectures do. The drawback of these architectures is that they do not provide mechanisms for holistic system configuration.

Our work is also inspired by recent work on energy management architectures for sensor networks [13,16]. Such architectures allow for applications to be configured to meet energy efficiency goals. The purpose of Chi, in contrast, is to provide a generalization of these principles that also extends to other objectives than energy-efficiency.

7 Conclusions

We present Chi, an architecture for full-system configuration and policy-based optimization in sensor networks. Unlike previous modular configuration architectures, Chi's dynamic properties make it possible to switch configuration policies and to add new parameters during run-time. Our experiments show that Chi improves the sensor network performance as much as specialized architectures, while maintaining a clear separation of concerns.

Acknowledgments

This work was partly financed by VINNOVA, the Swedish Agency for Innovation Systems, and by SSF. This work has been partially supported by CONET, the Cooperating Objects Network of Excellence, funded by the European Commission under FP7 with contract number FP7-2007-2-224053.

References

1. Buettner, M., Yee, G.V., Anderson, E., Han, R.: X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In: Proceedings of ACM SenSys 2006, Boulder, Colorado, USA (2006)
2. Burri, N., von Rickenbach, P., Wattenhofer, R.: Dozer: ultra-low power data gathering in sensor networks. In: Proceedings of ACM/IEEE IPSN 2007, Cambridge, Massachusetts, USA (2007)
3. Cheng, T.E., Fonseca, R., Kim, S., Moon, D., Tavakoli, A., Culler, D., Shenker, S., Stoica, I.: A modular network layer for sensornets. In: Proceedings of USENIX OSDI 2006, Seattle, Washington, USA (August 2006)
4. Corkill, D.: Blackboard systems. AI Expert 6(9), 40–47 (1991)
5. Dunkels, A., Österlind, F., He, Z.: An adaptive communication architecture for wireless sensor networks. In: Proceedings of ACM SenSys 2007, Sydney, Australia (November 2007)
6. Dunkels, A., Österlind, F., Tsiftes, N., He, Z.: Software-based on-line energy estimation for sensor nodes. In: Proceedings of EmNetS-IV, Cork, Ireland (June 2007)
7. 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 Sixth ACM Conference on Networked Embedded Sensor Systems (ACM SenSys 2008), Raleigh, North Carolina, USA, November 2008, pp. 421–422 (2008)
8. Hoesel, L.V., Nieberg, T., Wu, J., Havinga, P.: Prolonging the Lifetime of Wireless Sensor Networks by Cross-Layer Interaction. IEEE Wireless Communications 11(6), 78–86 (2004)
9. Hui, J., Culler, D.: IP is Dead, Long Live IP for Wireless Sensor Networks. In: Proceedings of the 6th international Conference on Embedded Networked Sensor Systems, Raleigh, North Carolina, USA (November 2008)
10. Jurdak, R., Baldi, P., Lopes, C.V.: Adaptive low power listening for wireless sensor networks. IEEE Transactions on Mobile Computing 6(8), 988–1004 (2007)
11. Kawadia, V., Kumar, P.: A cautionary perspective on cross-layer design. IEEE Wireless Communications 12(1), 3–11 (2005)

12. Kim, S., Fonseca, R., Dutta, P., Tavakoli, A., Culler, D., Levis, P., Shenker, S., Stoica, I.: Flush: A reliable bulk transport protocol for multihop wireless networks. In: Proceedings of ACM SenSys 2007, Sydney, Australia (November 2007)
13. Klues, K., Xing, G., Lu, C.: Link layer support for unified radio power management in wireless sensor networks. In: Proceedings of ACM/IEEE IPSN 2007, Cambridge, Massachusetts, USA (2007)
14. Köpke, A., Handziski, V., Hauer, J.-H., Karl, H.: Structuring the information flow in component-based protocol implementations for wireless sensor nodes. In: Proceedings of Work-in-Progress Session of EWSN 2004, Berlin, Germany (January 2004)
15. Lachenmann, A., Marrón, P., Minder, D., Gauger, M., Saukh, O., Rothermel, K.: TinyXXL: Language and runtime support for cross-layer interactions. In: Proceedings of IEEE SECON 2006, Reston, Virginia, USA (2006)
16. Lachenmann, A., Marrón, P., Minder, D., Rothermel, K.: Meeting lifetime goals with energy levels. In: Proceedings of ACM SenSys 2007, Sydney, Australia (2007)
17. Langendoen, K., Baggio, A., Visser, O.: Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. In: Proceedings of IEEE IPDPS 2006, Rhodes Island, Greece (April 2006)
18. Madan, R., Cui, S., Lall, S., Goldsmith, A.: Cross-layer design for lifetime maximization in interference-limited wireless sensor networks. In: Proceedings of IEEE INFOCOM 2005, Miami, Florida, USA (March 2005)
19. Musaloiu-E., R., Liang, C.-J.M., Terzis, A.: Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. In: Proceedings of ACM/IEEE IPSN 2008, St. Louis, Missouri, USA (2008)
20. Polastre, J., Hui, J., Levis, P., Zhao, J., Culler, D., Shenker, S., Stoica, I.: A unifying link abstraction for wireless sensor networks. In: Proceedings of ACM SenSys 2005, San Diego, California, USA (2005)
21. Voigt, T., Österlind, F.: CoReDac: Collision-free command-response data collection. In: Proceedings of IEEE ETFA 2008, Hamburg, Germany (September 2008)
22. Whitehouse, K., Sharp, C., Brewer, E., Culler, D.: Hood: a neighborhood abstraction for sensor networks. In: Proceedings of ACM MobiSys 2004, Boston, MA, USA (June 2004)
23. Woo, A., Tong, T., Culler, D.: Taming the underlying challenges of reliable multihop routing in sensor networks. In: Proceedings of ACM SenSys 2003, Los Angeles, California, USA (2003)
24. Ye, W., Silva, F., Heidemann, J.: Ultra-low duty cycle mac with scheduled channel polling. In: Proceedings of ACM SenSys 2006, Boulder, Colorado, USA (2006)

Virtualising Testbeds to Support Large-Scale Reconfigurable Experimental Facilities

Tobias Baumgartner[1], Ioannis Chatzigiannakis[2,3], Maick Danckwardt[4], Christos Koninis[2,3], Alexander Kröller[1], Georgios Mylonas[2,3], Dennis Pfisterer[4], and Barry Porter[5]

[1] Dept. of Computer Science, Braunschweig University of Technology, Germany
{tbaum,kroeller}@ibr.cs.tu-bs.de
[2] Research Academic Computer Technology Institute, Patras, Greece
{ichatz,koninis,mylonasg}@cti.gr
[3] Computer Engineering and Informatics Department, University of Patras, Greece
[4] Institute of Telematics, University of Lübeck, Germany
{danckwardt,pfisterer}@itm.uni-luebeck.de
[5] Computing Department, Lancaster University, UK
b.porter@lancaster.ac.uk

Abstract. Experimentally driven research for wireless sensor networks is invaluable to provide benchmarking and comparison of new ideas. An increasingly common tool in support of this is a testbed composed of real hardware devices which increases the realism of evaluation. However, due to hardware costs the size and heterogeneity of these testbeds is usually limited. In addition, a testbed typically has a relatively static configuration in terms of its network topology and its software support infrastructure, which limits the utility of that testbed to specific case-studies. We propose a novel approach that can be used to (i) interconnect a large number of small testbeds to provide a federated testbed of very large size, (ii) support the interconnection of heterogeneous hardware into a single testbed, and (iii) virtualise the physical testbed topology and thus minimise the need to relocate devices. We present the most important design issues of our approach and evaluate its performance. Our results indicate that testbed virtualisation can be achieved with high efficiency and without hindering the realism of experiments.

1 Introduction

Experimentally driven research for wireless sensor networks has been instrumental in advancing the state of the art in recent years; new sensing applications, network architectures and protocol stacks have been optimised to operate over varied radio technologies, restricted resources and specific deployment strategies. The most commonly applied technique is *simulation* which allows rapid development, offers debugging tools and enables easy repeatability. A natural step beyond this is to implement the system on real hardware platforms and perform experiments in controlled testbed environments. This allows researchers

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 210–223, 2010.

to escape the inherent limitations of simulation regarding the available hardware characteristics (e.g. buffer sizes, available interrupts) and communication technology behaviour (e.g. transmission rates, interference patterns).

In the majority of cases, due to the costs of hardware, researchers evaluate their solutions in local testbeds of limited size. While small testbeds provide useful insights into the effectiveness of the system in real conditions, they tend to offer limited support in terms of heterogeneity, scalability and mobility. Furthermore, in most cases, a tightly coupled network and software architecture is followed on a testbed, thus limiting the number of possible configurations of that testbed.

In order to overcome limitations in scale, a number of testbeds of significant size have been developed in the last few years. Their size currently ranges up to 1000 nodes, and there is a trend towards building *even larger* testbeds as seen by projects such as WISEBED [14] and SENSEI [10]. This trend continues to serve more accurate experimentation – and therefore high quality research – in realistically-sized networks towards the scales imagined by the initial vision of sensor networking that dealt with using thousands or even tens of thousands of nodes.

Given this clear and continuing need for large open testbeds in WSN research, certain critical questions are posed: i) how do we deal with the ever-increasing total-number-of-nodes demand, ii) how do we combine large testbeds with heterogeneity (in available sensors, radios, computational resources, etc.), iii) how can we maintain a very large WSN testbed efficiently? Furthermore, how can we cater for hybrid simulation approaches, i.e., the combination of real and simulated testbeds in order to produce extremely large-scale WSN testbeds? Moreover, how do we utilize the facilities provided by these testbeds and adapt them to each experiment's needs; i.e. how can we define and use specific network topologies that fit into our target application domain?

We argue here that an efficient and flexible answer to such problems is the use of *federated testbeds* that unite isolated WSN testbed "islands" with the use of a *virtual links* concept. We propose the use of virtualised network links in the following ways:

- *Between physically distinct testbeds* of varying features (location, size, etc.) as a whole, but also *between specific nodes* of such testbeds, resulting in larger testbeds with customised cross-network edges,
- *Between nodes inside a single testbed*, thus defining a customised network topology,
- *Between real and simulated nodes*, enabling hybrid simulation for massive network sizes.

A virtual link essentially enables two testbed nodes, that have otherwise no direct physical radio connection, to communicate in a way that is transparent to the user applications; additionally, existing 'links' (i.e. reachability within one-hop radio range) can be selectively deactivated between neighbouring nodes. Both kinds of virtualisation are done in a way that is entirely transparent to a deployed application.

The major challenge arising from using such an approach is in the extent to which this virtualisation affects the realism of the experiments conducted – the tradeoff between the ability to extensively scale and reconfigure testbeds in a straightforward way and its impact on the realism of results. In relation to this we currently target only experiments which use higher layers of network abstraction, avoiding those which operate at the the MAC layer. However, we believe that the "simulation" of network links and the resulting federated testbeds will prove itself largely beneficial to the research community.

In this work, we provide a systematic definition of our testbed virtualisation concepts, discussing in-depth design, architecture and implementation issues of our approach. We provide an evaluation of our work to demonstrate the feasibility of testbed virtualisation, comparing results from real network topologies against virtualised ones. Our results show that in many cases virtualisation can be efficiently integrated into testbeds without having a profound effect on the experimental results' realism.

This paper is structured as follows: a description of related work follows in Section 2. An in-depth discussion of our virtual link service follows in Section 3, with a set of experiments and results described in Section 4. This is followed by an example application (i.e., an experiment using virtual links) in Section 5. A discussion of our results is provided in Section 6.

2 Related Work

There is a significant body of existing work in the area of sensor network simulation and testbed infrastructures, with a number of works following a hybrid approach in the last few years; charasteristic examples of this approach are [7,8,3,12]. In such approaches part of the experiment is conducted in simulation and part on real hardware, with the ratio varying in different approaches. In some cases, only the wireless communication channel of the real devices is utilised with the rest of the software being executed inside a simulator. In other cases the software is executed iteratively on real and simulated devices with certain arbitration and timing schedules applied. These concepts are somewhat related to our own, but we aim at using virtual links between real and simulated devices *in real time and simultaneously*.

There is also significant work regarding WSN testbeds and their respective management and debugging software. Large testbeds such as Trio [2], Motelab [13], TWIST [4] or SIGNETLAB [1] are accompanied by software that provides users with the facilities to conduct experiments with the testbed nodes, but are generally limited in their adaptability and configurability to the users' needs. Capabilities such as reconfiguring the network topology, or federating multiple testbeds to form a larger virtualised facility are to our knowlegde not provided in these cases.

Virtualised network links or federated testbeds and their use in network testbeds are in themselves not a new concept, with projects such as Planetlab implementing similar concepts. Additionally one recent approach dealing

with similar issues is [5], where the infrastructure and software of the Kansei testbed is combined with that of GENI in order to provide a unifying solution for discrete testbeds. Also, the Senslab project [11] aims to unify 4 discrete heterogeneous testbeds into a single one of 1000 nodes. However, all of these works lack a generalised approach allowing arbitrary configurations between real and virtual. Overall, our approach aims at providing a unifying abstraction for discrete testbeds and increased flexibility in defining network topologies, without hindering the efficiency and realism of using a testbed versus pure simulation.

3 Virtual Links and Federated Testbeds

In this section we describe in detail our approach to virtualising testbeds. We define a virtualised testbed as either a single physical testbed with a virtualised topology; two or more physically distinct testbeds federated into a single unified testbed; a simulated testbed similarly federated with a physical testbed; or any combination of the above.

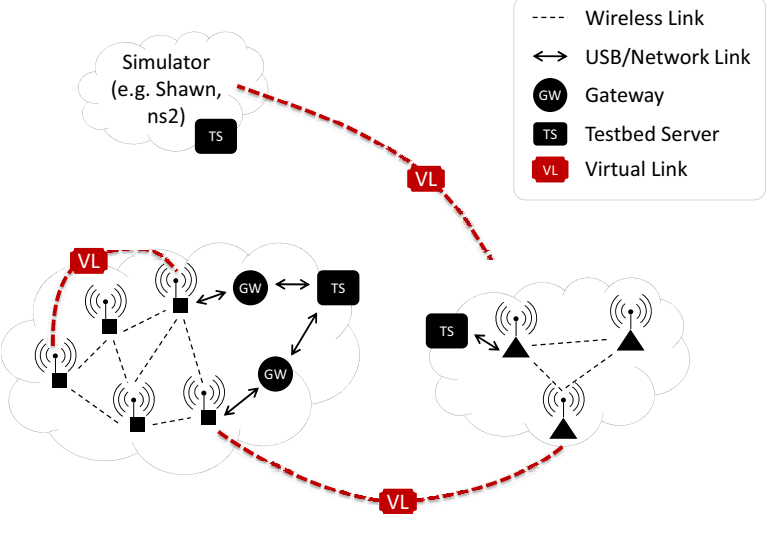


Fig. 1. The architecture of virtualised testbeds

The key components of our architecture are shown in figure 1. Each testbed – physical or simulated – is represented by a *testbed server* which acts as the Internet-facing gateway to the testbed. A testbed itself is composed of a number of *sensor nodes* which can communicate with the testbed server (potentially via gateway devices inside a physical testbed).

A *virtual link* is then a (unidirectional) connection between two nodes – in the same or in different testbeds – which would not normally be able to communicate.

An arbitrary number of virtual links can thus be created to define a virtualised topology and federate distinct testbeds. We can also *deactivate* existing physical reachability between two nodes by selectively dropping packets to allow complete topology control.

In more detail, virtual links are enabled with a special piece of software on each sensor node – a *virtual radio* – which contains a routing table of the form {ID, interface}, such that when sending a message to a specific node ID the radio can decide on which 'interface' to send this message; the node's real radio or the virtual interface which forwards the message to the testbed server (where it is routed onwards as appropriate).

In the remainder of this section we discuss the virtual radio software in detail and its interaction with a testbed server. Following this we describe the unified message format which allows heterogeneous nodes to communicate, we discuss the ways in which virtualised topology can support link quality modelling, and finally we describe how simulation is integrated into real-time testbed experiments.

3.1 Topology Virtualisation

Virtualising topology involves two key elements, *virtual radio* components, and *testbed servers*, shown in Figure 2.

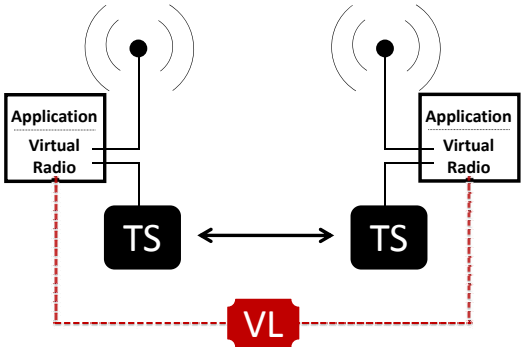


Fig. 2. The communication beween virtual radio drivers on sensor nodes

At the start of an experiment, the IDs of virtual radios across the entire virtualised topology are configured by an overall controlling component, ensuring uniqueness. Virtual topology itself is configured by each testbed server informing its local sensor nodes of their virtual neighbours, where a virtual neighbour entry in a sensor node's virtual radio simply consists of an ID along with 'virtual', meaning any packets to this ID should be sent to the testbed server for further routing. How messages reach the testbed server depends on the architecture of the deployed testbed – routing may be via an out of band backbone infrastructure when sensor nodes are connected 1:1 with gateway devices, or alternatively may

reuse the wireless medium of the sensor nodes in testbeds where not every sensor node is directly connected to a gateway device. In either case the procedure is transparent to the application software.

The process of sending a message thus works as follows: applications on a sensor node send a packet to its virtual radio component. On some operating systems (e.g., TinyOS), using a virtual radio instead of a real one is simply a matter of component configuration. On others, it may require changing radio function calls in the application's source. The virtual radio component then uses its local routing table (configured by the testbed server as above) to decide on which interface to send this message – via the real radio or the virtual topology service via the testbed server. When broadcasting, a packet is simply sent on both interfaces.

If the message is sent to the testbed server, the server examines the destination ID of the packet and forwards this either to another testbed server which is responsible for that node, or to the corresponding node in the local testbed. If the packet is broadcast, the testbed server forwards the message to all virtual neighbours of this node (i.e. generating one message for each neighbour).

Finally, when receiving a message on the *real* radio interface, the virtual radio component checks its routing table to determine whether or not the sender is configured to be a neighbour in the currently configured topology, and if not then the packet is simply dropped and so never reaches the application. All parts of this procedure can be completely transparent to the application, which can simply see a radio component conforming to a common radio API.

3.2 Message Format

Packets traveling over virtual links between testbed servers have a common format, and in cases where virtual links exist between nodes of different types, the local testbed server performs appropriate translation of the packet to a format suitable for use at the destination node. This process is also used when a single testbed has heterogeneous nodes – or nodes running different operating systems – with virtual links between them.

The common packet format therefore abstracts over different concepts of link quality between platforms (such as LQI or RSSI values) and other differences in the types of fields present in packets, different offsets for the same fields within packets, or different lengths of addresses. For this reason, we define a generic representation of a packet that is used when testbed servers forward messages from a virtual link between nodes. An example of a generic packet is shown below:

```
<?xml version="1.0" encoding="UTF-8">
<Node2Node_Packet>
     <sender_ID ID="urn:testbed1:node1" />
     <destination_ID ID="urn:testbed2:node2" isBroadcast="false" />
     <message>A1 DF 63 8B</message>
     <LQI>200</LQI>
     <RSSI>199</RSSI>
     <Options>ACK</Options>
</Node2Node_Packet>
```

This example contains the ID of the sender node (sender_ID), the ID of the destination node (destination_ID), whether the message is a broadcast, the payload of the message, an LQI value, an RSSI value and optional flags such as whether the sender needs an acknowledgment.

3.3 Modelling Link Characteristics

In addition to the basic routing functionality involved in virtualisation, testbed servers may perform traffic shaping on messages sent over virtualised links according to some desired model, for example emulating lossy channels or interference. Although beyond the scope of this paper, it is easy to plug such models into the software used at testbed servers.

3.4 Simulation Considerations

Connecting a simulator to a real testbed presents some unique challenges. Our motivation behind it is to enable ultra-large-scale experiments in which a large number of simulated nodes provide the macro-view of an experiment and serve as a test load to a relatively small number of real sensor nodes, on which the experiment outcome is measured. Using topology virtualisation the real nodes can be placed anywhere within the broader simulated topology.

Simulator integration follows the same implementation pattern as physical testbeds, i.e., the simulator is connected to a testbed gateway. A real-time enabled network simulator can be easily adapted to such an architecture. We chose the Shawn [6] network simulator for our experiments. The required modifications are also possible in other simulation environments.

Virtual link integration involves three major steps:

- Real-time simulation is required for a shared time basis between real and simulated nodes.
- Multi-threaded injection of messages into the simulation whenever a real sensor node (or alternatively a simulated node from another simulator) sends a message over a virtual link.
- Appropriate message 'routing' inside the simulator must be added, such that the simulator's testbed gateway routes messages sent from simulated to real nodes over its network connection.

This approach implies that the simulator is able to execute the simulation of nodes fast enough to keep up with real time; Shawn is a high-level simulator, and can easily do this for thousands of simulated nodes.

4 Evaluation

In this section we evaluate the *realism* of virtual links in experiments conducted over a federation of separate physical testbeds. Furthermore, we evaluate the

efficiency of our implementation in terms of *latency* (transmission delay of messages exchanged over virtual links) and *scalability* (increase in latency as the volume of messages over virtual links increases).

We implement the virtual radio component (see section 3.1) on three WSN operating systems: TinyOS, Contiki, and iSense. This makes our approach operable on a wide variety of hardware platforms such as the Crossbow mote series, Tmote Sky, ScatterWeb motes and iSense nodes. Our testbed server software is implemented in Java using Web Services for inter-server communication.

We evaluate the time to transmit a message over (i) the physical hardware radio, as a benchmark, (ii) a virtual radio using the UART to send virtual link messages to a directly connected gateway device (referred to as *Setup-I*), and (iii) a virtual radio implemented using the physical radio to forward virtual link messages to a gateway-connected sensor node (referred to as *Setup-II*).

We test the hardware radio in a simple two-node topology, providing a reference result for the speed of real radio messages. For the virtual radio tests we use two physically separate sensor nodes that are in different testbeds, in one case such that each sensor node is connected directly to a gateway device, and in a second case where each sensor node must use its hardware radio to forward virtual link messages to another sensor node which is directly connected to a gateway device – these represent the two most common kinds of WSN testbed deployments. In the virtual radio tests we have two testbed servers (one for each testbed) connected via gigabit Ethernet in the same LAN, thus simulating a full virtual link message transport procedure (as illustrated in figure 2).

For each type of link and each hardware platform considered we transmit a total of 1000 messages. Table 1 shows the minimum, maximum and average times taken for a message to be sent from an application at one sensor node and arrive at the other[1].

Table 1. The min/avg/max message transit times for 3 platforms using a physical radio, a virtual link over UART (Setup-I) and a virtual link over radio (Setup-II)

Hardware Platform	Physical Radio			Setup-I			Setup-II		
	min	avg	max	min	avg	max	min	avg	max
ScatterWeb	72.2ms	75ms	81.6ms	4.7ms	5ms	5.2ms	149.4ms	155.3ms	167.5ms
telosB	38.7ms	40.2ms	43.3ms	4.7ms	5ms	5.3ms	81.2ms	85.6ms	92.4ms
iSense	6.3ms	7ms	8.1ms	4.7ms	5ms	5.2ms	14.1ms	17.9ms	20.6ms

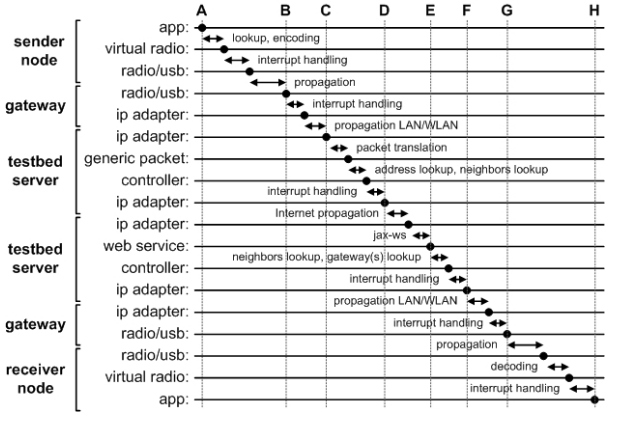
The results show significant variation in the message transit times when using the physical hardware radios of the different platforms; this is caused by differences in hardware design. In the iSense platform for example the JN5139 microcontroller integrates an on-chip CC2420 radio module thus increasing the

[1] To accurately measure the transmission time over the ScatterWeb platform we used a high-precision external clock, as the internal hardware clock has low accuracy.

speed of communication. By contrast, on the TelosB platform, the MSP430 controller has an external CC2420 radio module accessed via an SPI bus, and the ScatterWeb node has an 868Mhz radio with a 19.2 kbit/s data rate.

When we use virtual radio links sending messages via UART on the other hand we see that all platforms perform almost identically. This is because the UART modules used by the different hardware platforms conform to the same serial protocol and apply the same settings (115200 bps 8N1). When using virtual radio links sending messages via the physical radio modules, we of course see similar variations in transit times caused by the hardware differences noted above.

These results demonstrate that the transmission times using virtual radios over UART are in fact faster than using physical hardware radios for all hardware platforms considered, even though this involves transport through the testbed servers. This allows us to potentially do further processing on virtual radio packets (such as modelling link characteristics) before delivering them to the final device without impacting on realism, or to simply impose a delay on such packets to make them comparable to physical radio message transit times. Alternatively we could send virtual radio messages between testbeds over the Internet and still deliver them in reasonable time.



Period	Setup-I	Setup-II
A-B	2.06ms	6.93ms
B-C	0.4ms	0.4ms
C-D	0.01ms	0.01ms
D-E	0.14ms	0.12ms
E-F	0.01ms	0.01ms
F-G	0.39ms	0.4ms
G-H	2.05ms	6.93ms
Total	4.96ms	14.69ms

Fig. 3. The timing of a transmission for each layer invoked of the sender, receiver and intermediate nodes for two different physical testbed setups

To further explore the sources of delay in sending messages over virtual radio links we instrumented as many steps as possible in each stage of the transport procedure, shown in Figure 3, from the application on the sending sensor node to the gateway device (labelled stage A-B), processing inside the gateway device (B-C) and testbed server (C-D), and the reverse of this for the second testbed. We use the same two virtual radio link implementations as before; Setup-I being over UART and Setup-II over virtual link message transported via the physical radio to a gateway. The average delays were again calculated based on a total

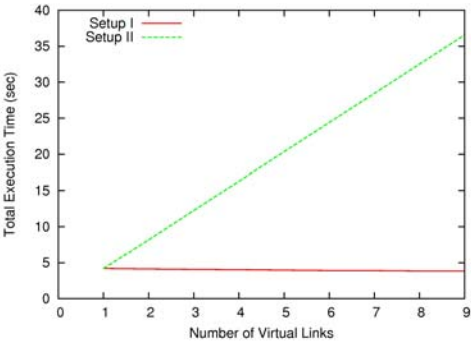


Fig. 4. Execution time for different number of virtual links on different testbed setups

of 1000 message exchanges for each setup considered. These results only show the iSense platform, since the periods A-B and G-H can otherwise be calculated from Table 1 for the other hardware platforms.

We conclude our performance evaluation by examining the scalability of our implementation. To do this we use a single testbed only with 10 sensor nodes. Nodes 1–9 are in one hop physical radio range of node 0 (the sink), to which they each transmit 1000 messages. We begin by using real hardware radio transmission for all nodes, then replace each of nodes 1–9's links to node 0 with virtual radio links. In each case we measure the total execution time for the sink to receive all 9000 messages. The results of this experiment are shown in Figure 4. The results indicate that testbed setup I, using UART for all virtual radio messages, is capable of delivering all 9000 messages over the virual links with almost no delay when compared to physical links. On the other hand, the total execution time of the experiment under testbed setup II, using the hardware radio to transport virtual radio messages via a single gateway device, linearly increases with the total number of messages transmitted over the virtual links. This is because in the latter case the single gateway node creates a bottleneck, and thus the total execution time increases with the number of messages transmitted over the virtual link (this is a worst-case scenario where only one gateway node exists).

5 Example Experiments Using Virtual Links

Having demonstrated the raw performance of our virtual radio implementations we now examine two characteristic WSN applications: sensor data aggregation and detection of a network partition. We execute these on testbeds using a mixture of real and virtualised topology, aiming to show that our approach does not impact the application's view of the network's behaviour – and therefore does not negatively affect results of experiments with these applications.

We use 3 different configurations: (i) a single testbed without any virtual links, (ii) two testbeds federated using virtual links whose testbed servers are on

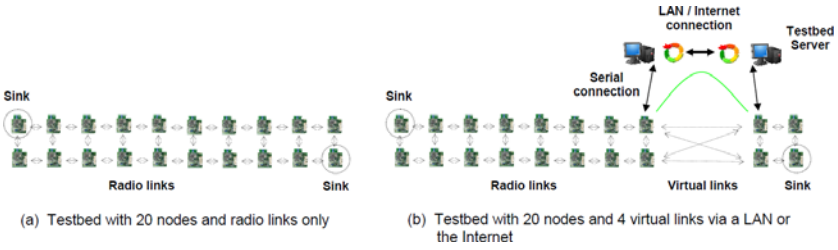


Fig. 5. Testbed Configuration

the same LAN, and (iii) two testbeds federated with virtual links whose testbed servers are at different locations within the same country and communicate over the Internet. Each configuration uses the same total number of nodes in the same topology, but in the virtualised cases some nodes are in different testbeds. For each configuration we evaluate how the applications perform to assess the effect of virtual links.

Sensor Data Aggregation. In this classic application, each node maintains an aggregate value (e.g. average) of a sensor reading (e.g. temperature) from all its neighboring nodes. Periodically all nodes report these aggregated values to the control center of the network (i.e. the sink). Each node broadcasts its sensor reading every 5sec. Nodes collect all data received and maintain the average value. Every 50sec each node sends the value to the sink using a simple flooding algorithm.

We deploy a single testbed without any virtual links (configuration I) that consists of 20 nodes arranged in two parallel lines of equal size with 1 meter distance between each node. The power output of radio interfaces is configured to achieve a maximum communication range of $1.5 - 2$m. The sink is placed at the top-left corner of the network (see Fig. 5a).

For the experiments with virtual links (configurations II and III), the testbed is separated in two parts consisting of 16 nodes and 4 nodes (Figure 5b). All nodes are directly connected to gateway devices via USB links (i.e. virtual radios using UART). We configure four virtual links between the border nodes of the two separated testbeds. In configuration II the two testbeds are within the same university LAN and are connected via a 100mbit Ethernet backbone. In configuration III the two testbeds are at different universities in the same country and are connected via the Internet. The connection between the two testbed servers in configuration III achieves an average ICMP echo request time of 12ms (over 100 echo requests) and a traceroute reported 15 hops.

For our evaluation we measure the average number of messages (i) sent by each node, (ii) received by each node via the actual radio component, (iii) received by each node via the virtual radio component and (iv) received by the sink for each 50sec cycle. We executed 10 experimental runs of 15min each for each testbed configuration. Table 2 shows the results for each metric over the three different testbed configurations when using iSense nodes (with similar results holding for

the other hardware platforms), demonstrating that the overall network behaviour in terms of messages received at the sink node is comparable in all cases, and general network behaviour in terms of messages sent and received is similar.

Table 2. Evaluation metrics for the Data Aggregation Example Experiment

Average number of messages	Configuration I	Configuration II	Configuration III
sent per node	373.53	377.95	386.95
received per node via real radio	1667.05	1085.47	954.95
received per node via virtual radio	–	771.25	806.75
received by sink per cycle	13.11	12.88	13.61

In performing these experiments, due to the speed of virtual links over real ones, it was necessary in configuration II to introduce a random delay of 1–3ms per virtual link message in order to match the behaviour of the physical radios; no such delay was imposed for configuration III, as the inherent delay of the national-level Internet link made the virtual links comparable to the real radio links. For comparison, using the ScatterWeb platform, with its slower hardware radio, we found random delays of 70–80ms (configuration II) and 50–65ms (configuration III) appropriate, while for the TelosB platform we used 32–40ms and 20–28ms respectively.

Partition Detection. Here we use the same network topology as in the previous application but position an additional sink at the other side of the network. In this application we wish to detect if the network is partitioned due to the failure of an intermediate node or due to a lossy radio link; in case of a partition we issue an alarm at the sink. To do this we implement the algorithm of [9], in which each sink essentially sends a beacon message every 5sec to the other sink by flooding it through the network. If a beacon message gets lost between the two sinks then an alarm message is generated.

For our evaluation we measured the average messages (i) sent by each node, (ii) received by each node via the real radio, (iii) received by each node via the virtual radio, (iv) sent by both sinks and (v) received by the sink for each 50sec cycle, with results shown in Table 3 (where partition alarms are occasionally raised due to lossy radio channels).

The results in Table 3 show that the network behaviour from the application's point of view (in terms of generated alarms) is comparable in both real and virtualised testbeds, and that network behaviour otherwise in terms of the numbers of messages sent and received is comparable.

Finally, table 4 shows how long it takes to send a partition beacon from one sink to the other and return to the initial sink. The roundtrip times here are almost identical for configurations I and II while for configuration III they are slightly higher due to latency introduced by the Internet link.

Table 3. Evaluation metrics for the Partition Detection Example Experiment

Average number of messages	Configuration I	Configuration II	Configuration III
sent per node	343.33	320.94	341.87
received per node via real radio	1395.45	870.45	857.05
received per node via virtual radio	–	611.51	687.76
sent in total by both sinks	355.12	361.60	360.32
Generated alarms (both sinks)	19.33	20.25	18.71

Table 4. Roundtrip times for one beacon message

	Configuration I	Configuration II	Configuration III
Minumum [ms]	46.66	42.24	49.39
Average [ms]	75.09	73.15	88.73
Maximum [ms]	110.99	100.51	139.22

6 Conclusion

Virtual network links, as defined and used in this work, are a means of easily reconfiguring and federating testbeds into large-scale networks with direct control over the topology. We have demonstrated that the approach is both realistic and performant enough for experiments at the higher network layers such that applications cannot distinguish between a fully real topology and a partially virtualised one. While we expect that experiments with MAC layer algorithms may reveal more subtle artifacts of virtualisation, and so would need care in deriving results, we believe that our approach is nonetheless highly beneficial in enhancing the utility of a single testbed beyond its fixed physical topology, in federating testbeds to enable extremely large scale experiments on real hardware and in enabling the integration of simulation for even larger networks.

When building federated testbeds for scale, we have shown that the interconnection of two closely located testbeds, i.e. with short delays between testbed servers, works very well in practice – as the virtual links operate significantly faster than physical links, an experiment can be tuned so that applications cannot detect that they are running in a physically separated network.

When moving to wider-area networks, we are limited by the existence of a sufficiently fast Internet backbone. While connections over 15 hops in a university-connecting national network fulfill such a requirement, we acknowledge that latency may degrade and affect realism too much in large intercontinental federations. However, we believe that future high-speed broadband networks will make this issue diminish in time.

We conclude that topology virtualisation is a promising approach to create large federations of physically separated and heterogenous networks. Building nationwide testbeds is achievable with today's technology, and we believe that worldwide networks are viable in the near future.

Acknowledgments

This work has been partially supported by the European Union under contract numbers IST-2008-224460 (WISEBED). The authors would like to thank Geoff Coulson, Sándor Fekete, Stefan Fischer, Qasim Mushtaq and Paul Spirakis for their insightful comments and ideas.

References

1. Crepaldi, R., Friso, S., Harris III, A.F., Zanella, A., Zorzi, M.: The design, deployment, and analysis of signetLab: a sensor network testbed and interactive management tool. In: WiNTECH 2006: Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization, pp. 93–94. ACM, New York (2006)
2. Dutta, P., Hui, J., Jeong, J., Kim, S., Sharp, C., Taneja, J., Tolle, G., Whitehouse, K., Culler, D.: Trio: enabling sustainable and scalable outdoor wireless sensor network deployments. In: Proceedings of the 5th international conference on Information processing in sensor networks (IPSN 2006), pp. 407–415. ACM, New York (2006)
3. Girod, L., Ramanathan, N., Elson, J., Stathopoulos, T., Lukac, M., Estrin, D.: Emstar: A software environment for developing and deploying heterogeneous sensor-actuator networks. ACM Trans. Sen. Netw. 3(3), 13 (2007)
4. Handziski, V., Köpke, A., Willig, A., Wolisz, A.: TWIST: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In: REALMAN 2006: Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality, pp. 63–70. ACM, New York (2006)
5. Wiki for the Kansei GENIE project, http://sites.google.com/site/siefastgeni/
6. Kröller, A., Pfisterer, D., Buschmann, C., Fekete, S.P., Fischer, S.: Shawn: A new approach to simulating wireless sensor networks. In: Design, Analysis, and Simulation of Distributed Systems 2005 (DASD 2005), April 2005, pp. 117–124 (2005)
7. Österlind, F., Dunkels, A., Voigt, T., Tsiftes, N., Eriksson, J., Finne, N.: Sensornet checkpointing: Enabling repeatability in testbeds and realism in simulations. In: Roedig, U., Sreenan, C.J. (eds.) EWSN 2009. LNCS, vol. 5432, pp. 343–357. Springer, Heidelberg (2009)
8. Park, S., Savvides, A., Srivastava, M.B.: Sensorsim: a simulation framework for sensor networks. In: MSWIM 2000: Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems, pp. 104–111. ACM Press, New York (2000)
9. Ritter, H., Winter, R., Schiller, J.: A partition detection system for mobile ad-hoc networks. In: Sensor and Ad Hoc Communications and Networks, IEEE SECON 2004, October 4-7, pp. 489–497 (2004)
10. SENSEI project website, http://ict-sensei.org/
11. Very large scale open wireless sensor network testbed, http://www.senslab.info
12. Wen, Y., Zhang, W., Wolski, R., Chohan, N.: Simulation-based augmented reality for sensor network development. In: SenSys 2007: Proceedings of the 5th international conference on Embedded networked sensor systems, pp. 275–288. ACM, New York (2007)
13. Werner-Allen, G., Swieskowski, P., Welsh, M.: Motelab: A wireless sensor network testbed. In: Fourth International Conference on Information Processing in Sensor Networks (IPSN 2005). IEEE, Piscataway (2005)
14. WISEBED project website, http://www.wisebed.eu/

Mitigating the Effects of RF Interference through RSSI-Based Error Recovery*

Jan-Hinrich Hauer, Andreas Willig, and Adam Wolisz

Telecommunication Networks Group
Technische Universität Berlin, Germany
{hauer,willig,wolisz}@tkn.tu-berlin.de

Abstract. On a common sensor node platform (Telos) we sample RSSI with high frequency *during packet reception*. We find that a packet collision (RF interference) often manifests as a measurable, temporal increase in RSSI. We investigate how the receiver can use this information to detect interference and, through temporal correlation, estimate the bit error positions in a corrupted packet. In an experimental study in two testbeds and several realistic BAN scenarios we show that a simple threshold-based algorithm often succeeds in estimating a large fraction of the bit error positions correctly. We develop an ARQ scheme that utilizes the error estimates to reduce the size of retransmitted packets. For this ARQ scheme we present an analytical model and verify it experimentally. Our results indicate that in comparison with a standard Send-and-Wait ARQ the expected number of bits per transmission can be reduced significantly (in our measurements by up to 14.7 %).

Keywords: Interference Mitigation, Packet Combining, ARQ.

1 Introduction

On a wired transmission medium the transmitter may be able to detect a collision at the receiver by monitoring the medium during the transmission, e.g. for an abnormal change in voltage. RF transceivers are usually half-duplex and the SNR conditions at transmitter and receiver can differ greatly. Therefore, transmitter-side collision detection schemes are generally not applicable in wireless communication. However, RF transceivers can often measure the power of the received radio signal and provide a corresponding RSSI (Received Signal Strength Indication), which may be used in the process of link quality estimation [12]. Previous work in low-power wireless networking has shown that a packet collision (RF interference) distorts the received signal and typically manifests as an additive increase in RSSI [7]. Consequently, RSSI has proven a relevant parameter when identifying RF interference as the cause of packet loss [5,10]. In this paper we

* This work has been partially supported by the European Commission under the contract FP7-2007-IST-2-224053 (CONET) and by the German Federal Ministry of Education and Research (BMBF) under the contract 01BN0712D (AVS-ZESAN).

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 224–239, 2010.

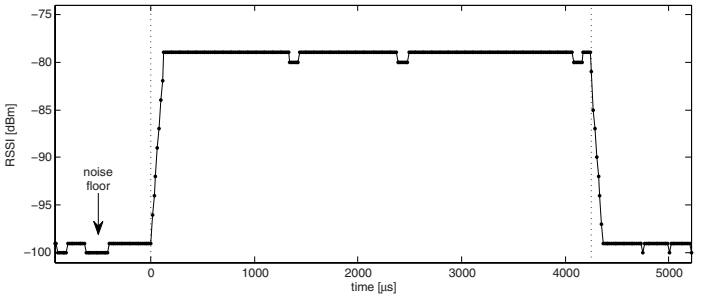


Fig. 1. RSSI sampled on a Tmote Sky with 62.5 kHz while receiving a maximum-sized IEEE 802.15.4 frame (133-byte PPDU) with an airtime of 4.256 ms

explore to what extent monitoring RSSI with high sampling rate *during packet reception* can be useful not only for receiver-side interference detection, but also for estimating the bit error positions inside corrupted frames.

On an IEEE 802.15.4-compliant radio platform (TI CC2420 [13]) we obtain an RSSI time series – a sequence of RSSI data points – by sampling RSSI with 62.5 kHz while receiving a frame (Fig. 1). We call such a time series the *RSSI profile* of the incoming frame. Our measurement results indicate that the variance of the RSSI profile of a correctly received frame is often much lower than that of a corrupted frame. When we take a closer look at corrupted frames we often find elevations in the RSSI profile whose duration matches the airtime of a colliding frame. And when we compare the frame payload with its RSSI profile we find that an increase in the RSSI profile often corresponds in time with the beginning of a segment of erroneous bits and a decrease marks its end.

We propose to use RSSI profiles for estimating the bit error positions inside corrupted frames. Our approach is applicable when errors are caused by RF interference (and elevations are observable in the RSSI profile), rather than by an insufficiently strong signal. With an ever increasing number of wireless devices, however, interference is becoming a major concern, especially in the ISM bands. We explore how RSSI profiling can help to improve the performance of Automatic Repeat reQuest (ARQ) protocols [4] in the presence of RF interference. Our approach belongs to the class of packet combining schemes [2]: when a corrupt frame is received, only the estimated erroneous portion is retransmitted, thus saving energy and bandwidth and reducing the probability of yet another error.

The rest of this paper is structured as follows: in Sect. 2 we present results from a set of baseline measurements investigating the dynamics of RSSI profiles during controlled collisions and in several environments of realistic, uncontrolled RF interference. Sect. 3 introduces and evaluates a simple, threshold-based algorithm that estimates bit error positions with the help of RSSI profiles. In Sect. 4 we present an analytical model and empirical results that show the performance improvements of an ARQ scheme when it is coupled with the algorithm. After an overview of related work in Sect. 5 the paper is concluded in Sect. 6.

2 Baseline Measurements

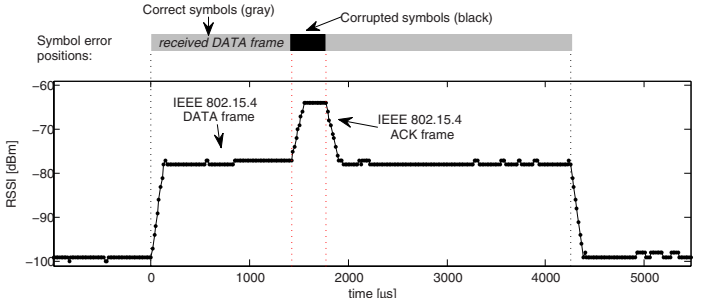
In this section we first report on some representative RSSI profiles that we obtained during controlled collisions with different types of 802.15.4 and WLAN frames. We then present results from several uncontrolled, realistic RF interference environments. All our experiments are performed with Telos (Tmote Sky [11]) sensor nodes, which are equipped with the IEEE 802.15.4-compliant Texas Instruments CC2420 transceiver [13].

2.1 CC2420

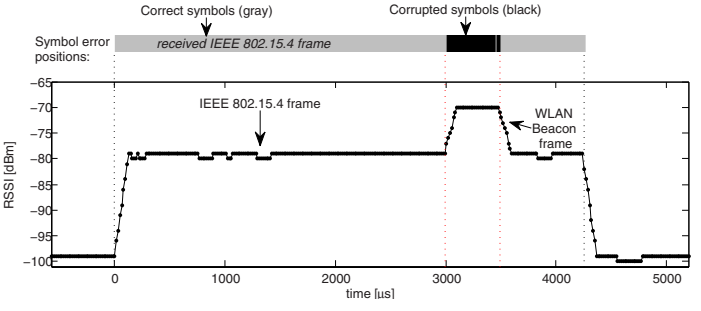
The CC2420 [13] operates in the 2.4 GHz ISM band at a nominal data rate of 250 kbps. A packet can be transmitted on one of 16 channels which are spaced 5 MHz apart and occupy 2 MHz of bandwidth. By default the radio automatically attaches to every received packet an RSSI value, which represents the average signal strength during packet reception. In conformance with the IEEE 802.15.4 standard the CC2420 allows to obtain the current RSSI by software. It is continuously updated and averaged over the last 8 symbol periods (128 μs). This allows to measure ambient RF noise, for example during an IEEE 802.15.4 energy detection scan, but also to obtain an RSSI sample *during frame reception*. As shown in Fig. 1, by reading RSSI with high frequency — we always use 62.5 kHz, the reciprocal of the time of a symbol — one can thus obtain the RSSI profile of an incoming frame as a moving average over a window of 128 μs. In practice, the sampling of the RSSI profile of an incoming frame is triggered by the reception of its start-of-frame delimiter (SFD), which results in an interrupt on the MCU. The end of an RSSI profile is marked by the reception of its last byte. Due to unknown interrupt service latencies the sampling routine continuously polls a radio pin for the end of the frame, because this point in time is used as reference to align the RSSI profile with the frame tail. Thus, in contrast to the illustrative results shown in Fig. 1 and 2, RSSI profiles do not include samples of the noise floor.

2.2 Controlled Collisions

In an environment of negligible external RF interference, which we verify with the help of a portable *Wi-Spy 2.4x* USB spectrum analyzer, we create controlled collisions between different types of WLAN and 802.15.4 frames. The basis for our measurement are two Tmote Sky sensor nodes: one periodically transmits maximum-sized 802.15.4 frames (133-byte PPDU with pre-defined content), the other listens for incoming frames while it continuously measures RSSI with a rate of 62.5 kHz. Whenever a frame (with a correct or incorrect CRC) is received, it outputs the frame and its RSSI profile over USB to a laptop. This allows us to analyze the dynamics in RSSI during reception and compare it with the bit error positions in a corrupted frame.



(a) Collision with an 802.15.4 ACK (11 byte PPDU).



(b) Collision with a WLAN beacon transmitted at 2 Mbps.

Fig. 2. Error positions and RSSI profiles of an IEEE 802.15.4 frame (133-byte PPDU) colliding with an IEEE 802.15.4 ACK (top) and IEEE 802.11 BEACON (bottom). The RSSI profile is measured on the Tmote Sky while receiving the frame.

In a first experiment we use a third Tmote Sky node to act as interferer: it periodically transmits small 802.15.4 DATA or ACK frames without clear channel assessment (CCA) and thus provokes collisions with the frames exchanged between the two other nodes. In a second experiment we replace the Tmote Sky interferer with a laptop which uses its WLAN PC card, based on an Atheros AR5212 chipset, to inject different types of WLAN frames (Beacon, ACK, RTS, CTS or DATA). WLAN frames are also sent without CCA and the channels are chosen such that they have maximum overlap. In all experiments the distances between the nodes/WLAN interferer are small ($< 5\,\mathrm{m}$) and the interferer is located closer to the receiver than the transmitter. Two representative results can be seen in Fig. 2.

The results show that (1) collisions are clearly visible by elevations in the RSSI profile, (2) the duration of an elevation matches the airtime of the colliding frame and, most importantly, (3) the positions of the bit errors in the received frame are temporally correlated with the elevation. Note that whenever there was no frame error (collision) the RSSI profile was typically stable ($\pm$ 1 dBm) as shown in Fig. 1.

2.3 Uncontrolled RF Interference

The previous results were obtained with an artificial setup. In this section we want to address three issues that might exclude the applicability of RSSI profiling in practice. They are related to the following questions:

1. How often does the radio hardware discard corrupted frames before they can be processed and potentially recovered by software?
2. When a corrupted frame is received, how many bits are erroneous?
3. Is there a substantial difference between the RSSI profile of a corrupted and a correctly received frame?

If only a small fraction of the corrupted frames is decoded and forwarded by the radio hardware; if in a corrupted frame typically most bits are erroneous; or if RSSI profiles are very similar regardless of a frame being corrupted or correct, then an error recovery scheme based on RSSI profiling is likely to be of little practical use. In the rest of this section we deal with these questions by performing a set of measurements in three different environments of uncontrolled, realistic RF interference.

The first two measurements are conducted in two publicly accessible indoor sensor network testbeds: TWIST [15] and MoteLab [16]. Both testbeds are situated on a university campus and have several WLANs co-located. Each testbed contains a large number of Tmote Sky nodes and we program one of them, located close to the geographic center of the testbed, to broadcast a total of 100,000 frames on channel 21[1], one frame every 125 ms. All other nodes listen for incoming frames and output every received frame and its RSSI profile over USB to be stored in a trace file for our later examination. Frames have an MPDU size of 64 byte and are transmitted with CCA: when the sender determines a busy channel the transmission is delayed for a small random time interval. Since we only use one sender a busy channel can only be caused by other users of the spectrum, e.g. WLAN. Each measurement lasts for about 4 hours and was, in both cases, performed on a weekday in the daytime.

For a third measurement we strap two Tmote Sky nodes to a person: one to the left upper arm, the other to the shin just above the ankle, resulting in a relative distance of about 1.5 m. One node is transmitting a total of 10,000 frames on channel 21 with a transmission power of -25 dBm, the other node listens and forwards received frames and their RSSI profiles over USB to a laptop carried in a backpack. Again, the MPDU size is 64 byte and frames are transmitted with CCA. In all experiments based on this body area network (BAN) setup the test person is walking outdoors on urban streets: a central urban shopping street and streets in a central residential area. We measure the same metrics as described in the previous paragraph, but in contrast to the testbed setups one experiment evaluates only a single link (we make a total of 6 BAN experiments).

Per setup we calculate for every link, i.e. sender-receiver pair, packet reception rate (PRR) as the ratio of correctly received to transmitted packets, and

[1] Channel 21 overlaps with (the popular) WLAN channel 10.

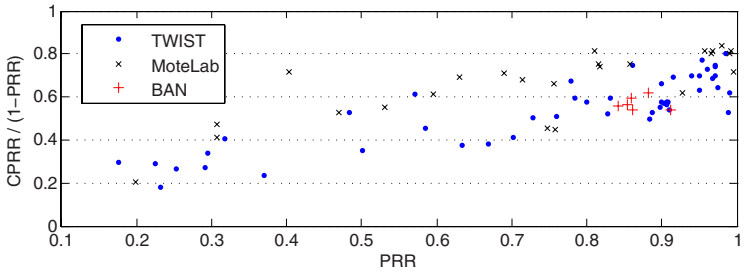


Fig. 3. Each point represents an 802.15.4 link over which 100,000 (TWIST/MoteLab) or 10,000 (BAN) frames were transmitted. The x-axis shows the link's PRR, the y-axis represents what fraction of those frames that were not received correctly was still accessible by the receiver MAC protocol.

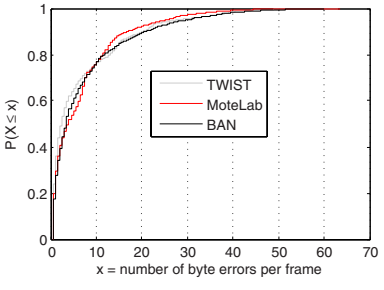


Fig. 4. Empirical CDFs for the number of byte errors per corrupted frame

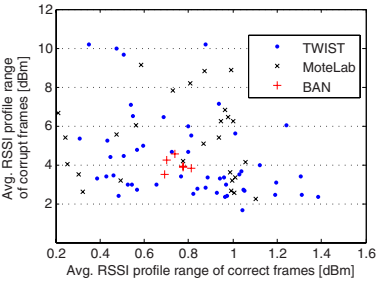


Fig. 5. Average range of the RSSI profile per link for correct vs. corrupt frames

corrupt packet reception rate (CPRR) as the ratio of received corrupt packets (CRC incorrect) to transmitted packets. Thus $CPRR/(1-PRR)$ describes what fraction of those frames that were not received correctly was still accessible by software. The remaining frames were discarded by the receiver radio supposedly due to a weak signal and/or errors in the PHY header. In our evaluation we only consider links that had a PRR between 0.1 and 0.995 and we only count as corrupt packets those that have the expected length.

Fig. 3 shows per-link $CPRR/(1-PRR)$ for all setups. On bad links sometimes only 20 % of those frames that were not received correctly were accessible by software, on good links typically at least 50 %. One possible explanation is that on bad links signal strength is often below the radio sensitivity threshold and frames are lost due to an insufficiently strong signal. The number of byte errors per corrupted frame is shown in the empirical CDFs in Fig. 4: on average no more than 5 bytes were incorrect, i.e. less than 10 % of the 64 byte MPDU. Finally, for every received packet we calculated the range over the 128 samples (2 samples correspond to one byte) contained in its RSSI profile, i.e. the interval between the minimum and maximum RSSI value. We then compared the average range for all correctly received vs. corrupted packets per link. The results are shown

in Fig. 5 and suggest that typically the RSSI profile of a frame that had at least one bit error had a considerably higher range than the RSSI profile of a frame that was received correctly.

In summary, the results indicate that (1) often a large portion of the corrupted frames is accessible and thus potentially recoverable, (2) the average number of erroneous bytes compared to a medium-sized frame is small and (3) RSSI profiles might contain enough information to estimate the bit error positions correctly. In the next section we investigate the last criterion more closely.

3 Estimating Bit Error Positions With RSSI Profiles

In this section we examine how to estimate the bit error positions in a corrupted frame based on its RSSI profile. We call an algorithm that performs this task a REPE algorithm (**R**SSI-based bit **E**rror **P**ostion **E**stimation). A REPE algorithm is invoked on the receiver upon reception of a frame with an incorrect checksum (CRC). It takes the RSSI profile of the corrupted frame as input and tries to output an estimate of the bit error positions. Note that in some cases the algorithm will not be able to output a decision, for example when the RSSI profile does not contain enough variance.

A good REPE algorithm will maximize the number of decisions, while minimizing the number of false positives (bits classified as incorrect, although they are correct) and false negatives (bits classified as correct, although they are incorrect). It should also impose little computational and memory overhead. In the rest of this section we introduce a simple REPE algorithm and evaluate its performance based on the traces we collected in the previous measurements.

3.1 A Threshold-Based REPE Algorithm

The REPE algorithm we propose simply marks all symbols that were received while the RSSI was above a certain threshold as incorrect. The threshold is defined relative to the RSSI of the incoming 802.15.4 frame, which we denote as $RSSI_{Base}$. In our scheme $RSSI_{Base}$ is set to the minimum value of the frame's RSSI profile, because previous work has shown that interference (collisions) results an RSSI increase rather than a decrease [7]. To exclude errors caused by an insufficiently strong signal the algorithm does not output a decision if $RSSI_{Base}$ is below the radio's sensitivity. Before the RSSI rises above and after it falls below the threshold an additional (temporal) safety margin is added. All bits between and including the safety margins are marked as incorrect and there can be multiple such sections per frame. Fig. 6 shows a pseudocode representation of the algorithm and visualizes its notation using an example.

The time complexity of the algorithm is $\mathcal{O}(n)$, where n is the number of samples in the RSSI profile. The memory requirements are n byte to store the RSSI profile — after the REPE algorithm has made a decision the same memory can be used to buffer the corrupted frame.

Algorithm. REPE-THRESHOLD($RSSI_{Profile}, \Delta_{threshold}, \Delta_{front}, \Delta_{rear}, sensitivity$)

$i \leftarrow 0$
$RSSI_{Base} \leftarrow \mathbf{min}(RSSI_{Profile})$
$symbolErrorMask[length(RSSI_{Profile})] \leftarrow \{0\}$
if $RSSI_{Base} \geq sensitivity$
then
  while $i < length(RSSI_{Profile})$
  do
    if $RSSI_{Profile}[i] \geq RSSI_{Base} + \Delta_{threshold}$
    then
      $first \leftarrow \mathbf{max}(i - \Delta_{front}, 0)$
      while $(i < length(RSSI_{Profile})$ **and**
        $RSSI_{Profile}[i] \geq RSSI_{Base} + \Delta_{threshold})$
        do $i \leftarrow i + 1$
      $last \leftarrow \mathbf{min}(i + \Delta_{rear}, length(RSSI_{Profile}) - 1)$
      $symbolErrorMask[first..last] \leftarrow 1$
    else $i \leftarrow i + 1$
return $(symbolErrorMask)$
comment: an empty *symbolErrorMask* (all zeros) means "no decision".

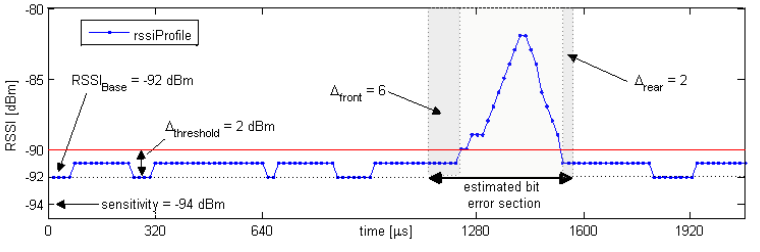


Fig. 6. The proposed REPE algorithm in pseudocode and an illustration of its notation

3.2 Evaluation

We evaluated the proposed REPE algorithm "offline", with the help of the traces we collected in the measurements described in Sect. 2.3. For every received frame our traces contained the RSSI profile as well as the content of the frame. From the latter we could infer the actual bit errors — the "ground truth". The algorithm was instantiated with parameters similar to the ones used in the example shown in Fig. 6: $\Delta_{threshold} = 2\,\text{dBm}$, $\Delta_{front} = 6\,\text{symbols}$, $\Delta_{rear} = 2\,\text{symbols}$ and $sensitivity = -93\,\text{dBm}$.[2] For every corrupted frame we let the REPE algorithm try to estimate the bit error section(s) and compared the result with the ground truth. The estimate was correct, if it resulted in no false negative, otherwise it was incorrect. In order to prevent trivial estimates (all bits marked as incorrect) an estimate that in total contained more than half of the MPDU size, i.e. more

[2] These parameters were chosen on the basis of a few trials - a more thorough investigation of the parameter space is part of our future work.

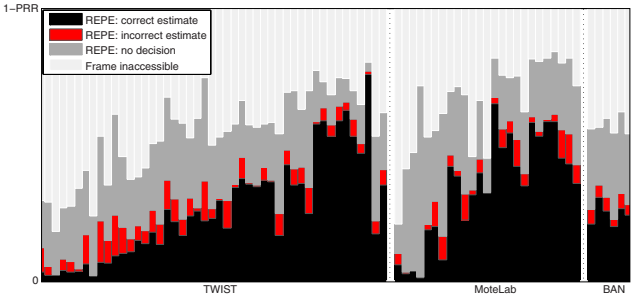


Fig. 7. Each bar represents an 802.15.4 link in one of the measurement setups. A bar shows for what fraction of those frames that were not received correctly the REPE algorithm made a correct, an incorrect or no decision, and what fraction was inaccessible by software. The bars are ordered by increasing PRR and each bar is normalized to 1-PRR (the PRR values can be extracted from Fig. 3).

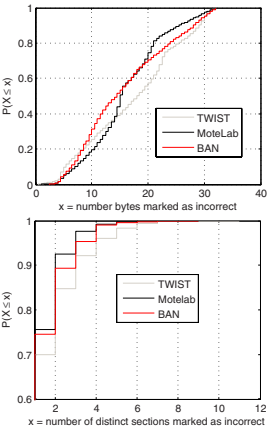


Fig. 8. Empirical CDFs for the errors estimates

than 32 byte, was treated as "no decision". We evaluated every link separately and again we only considered links that had a PRR between 0.1 and 0.995.

Fig. 7 shows how the REPE algorithm performed for each link in the three setups. Every link is represented by a bar that is subdivided into four parts, reflecting what happened to those frames that were not received correctly: they were either not accessible by software, or the REPE algorithm made either a correct or incorrect estimate, or it could not make a decision at all. The bars are ordered by increasing PRR and each bar is normalized to $1 - PRR$. It can be seen that the results exhibit strong variance, but there is also a clear trend: on bad links (low PRR, i.e. left side, respectively) typically the vast majority of frames that were not received correctly was either not accessible by software or the REPE algorithm could not output a decision. Also, at low PRR the number of incorrect estimates sometimes outnumbered the correct ones. With growing PRR, however, the proportion of REPE decisions increased to 50% and above, and for $PRR > 0.7$ the ratio of correct to incorrect decisions was typically at least 4. Again, a possible explanation for this trend is that on bad links packet loss is often also caused by a weak signal rather than (only) RF interference.

Fig. 8 shows CDFs for the number of bytes and distinct sections marked as incorrect by the REPE algorithm. With 15-20 bytes the average number of bytes classified as incorrect is about 4-5 times higher than the actual average number of byte errors (c.f. Fig. 4). This suggests that the proposed REPE algorithm might still be improved in terms of false positives, potentially by better parameter tuning or more fine-grained information from the radio hardware (e.g. shorter RSSI time windows or per-symbol LQI/correlation values). According to the bottom graph of Fig. 8 the estimated errors were often confined to a single section and in about 90% of all cases no more than two sections were identified.

These results confirm the general applicability of the algorithm, in the next section we examine one practical application more closely.

4 REPE-ARQ

There are several application areas that could benefit from the coupling with a REPE algorithm, for example Forward Error Correction (FEC) schemes or techniques that identify/exclude a certain type of interferer technology based on the airtime/inter-frame spacings of colliding frames [1]. In this section, however, we explore how a REPE algorithm can improve Automatic Repeat reQuest (ARQ) protocols [4].

In the standard Send-and-Wait ARQ scheme a frame is retransmitted when a bit error has occurred, typically noticed at the transmitter by the absence of an acknowledgement. We extend this scheme as follows: when a corrupted frame (CRC incorrect) is received, the REPE algorithm is invoked on the receiver side. If the algorithm can make an error estimate, the corrupted frame is buffered and a negative acknowledgement (NACK) frame is transmitted. As depicted in Fig. 11 a NACK is similar to an IEEE 802.15.4 ACK frame, except that it contains at least four bytes of payload, denoting the byte-offset and length of the estimated corrupted section(s) in the received frame, and a 16-bit REPE CRC. The number of *REPE offset/length* fields is variable to account for multiple error sections — alternatively a bitmask could be used to specify the error estimates, but it would typically impose more overhead since the number of error sections is expected to be small (c.f. Fig. 8). The REPE CRC is calculated over the *remaining* presumably uncorrupted portion of the received frame and allows the transmitter to determine whether the REPE algorithm on the receiver side was successful: it simply calculates the CRC over the original frame less the section(s) specified by the REPE offset and REPE length, and compares the result with the REPE CRC (all contained in the NACK). If they match, the receiver's estimate was correct, i.e. there was no error outside the estimated corrupted section[3], otherwise it was incorrect. In the first case only the corrupted portion (plus a 3-byte 802.15.4 MAC header) is retransmitted, indicated by an unused bit-flag in the header, which we call the `R-Flag` (we use bit 7 of frame control field). Upon correct reception the receiver is then able to assemble the original frame, pass it to the next higher layer and transmit a final ACK. In case this ACK is not received, the transmitter retransmits the reduced-size frame. If the transmitter discovers that the estimate was incorrect the entire frame is retransmitted. A flowchart of the REPE-ARQ scheme is shown in Fig. 9 (receiver-side) and 10 (transmitter-side).

4.1 Analytical Model

We use a time-homogeneous discrete-time Markov chain model [8] to analyze the performance of the REPE-ARQ algorithm with a maximum of k allowable

[3] Like the IEEE 802.15.4 MAC we ignore the residual error rate of the 16-bit CRC.

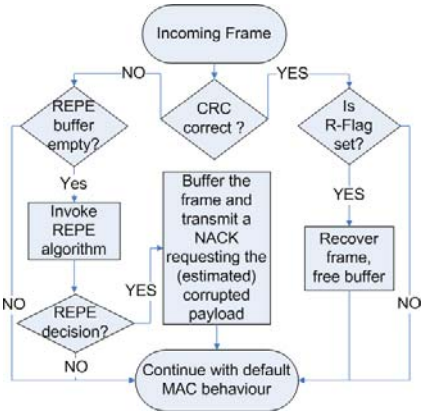


Fig. 9. The REPE-ARQ scheme (receiver-side)

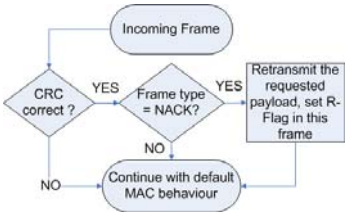


Fig. 10. The REPE-ARQ scheme (transmitter-side)

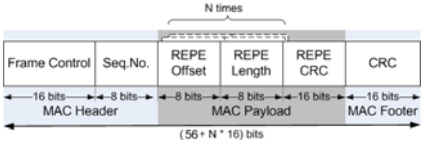


Fig. 11. NACK frame format

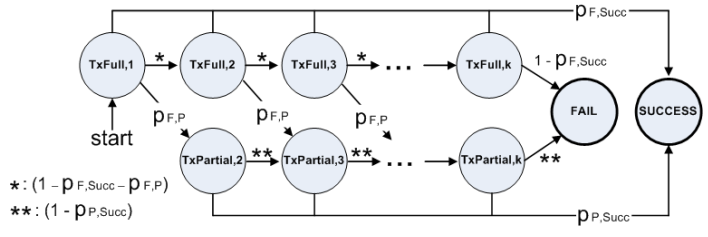


Fig. 12. The Markov chain model for the REPE-ARQ scheme

trials. This model is based on the assumption that different trials to transmit a frame are independent of each other.

The model's state transition diagram is shown in Fig. 12. From this diagram the underlying state transition matrix $\mathbf{P}$ can be directly derived. In the `TxFull,i` states (where i represents the number of trials) the transmitter sends a full-sized packet, whereas in the `TxPartial,i` states it sends a partial packet, i.e. retransmits only those parts of the payload that were previously corrupted and correctly identified as such by the REPE algorithm. In the `SUCCESS` state the transmitter has received an acknowledgement, whereas in the `FAIL` state the transmitter has exhausted all k allowable trials without receiving an acknowledgement. A transition from `TxFull,i` to `SUCCESS` occurs when a full-sized packet and its ACK were received without errors (with probability $p_{F,Succ}$); a transition `TxFull,i` to `TxPartial,i+1` occurs when a full-sized packet with errors was received, the REPE algorithm made a correct estimate and the NACK was received without errors (with probability $p_{F,P}$); and a transition `TxPartial,i` to `SUCCESS` occurs when a partial packet and its ACK were received without errors (with probability $p_{P,Succ}$).

The remaining state transition probabilities can be derived from the three basic probabilities $p_{F,Succ}$, $p_{P,Succ}$ and $p_{F,P}$ in a straightforward way. They are shown in Fig. 12. To instantiate the model we obtain the three basic probabilities from measurements (c.f. Sect. 4.3).

Our goal is to compute the average total number of bits transmitted within the k possible trials. To achieve this, we utilize the framework of potential theory for Markov chains [8, Sec. 4.2], the relevant definitions and theorems are paraphrased in Appendix A. In our case, all states `TxFull,i` and `TxPartial,i` are inner states, whereas the absorbing states `FAIL` and `SUCCESS` are final states. In the final states no transmission costs are incurred, all states `TxFull,i` have the same cost c_F and all states `TxPartial,i` have the cost c_P.

The average number of transmitted bits in the `TxFull,i` states, c_F, consists of the following components:

- the number of bits in a full-sized data frame, $l_{D,F}$,
- the number of bits in a positive acknowledgement, l_{ACK}, weighted with probability $p_{F,Succ}$, and
- the average number of bits in a negative acknowledgement, l_{NACK}, weighted with probability $p_{F,P}$.

Summarizing:

$$c_F = l_{D,F} + p_{F,Succ} \cdot l_{ACK} + p_{F,P} \cdot l_{NACK}$$

Similarly:

$$c_P = l_{D,P} + p_{P,Succ} \cdot l_{ACK}$$

where $l_{D,P}$ is the average number of bits for a partial data frame.[4] The average number of transmitted bits is computed by solving $\phi = \mathbf{P} \cdot \phi + \mathbf{c}$ for ϕ and reading off from ϕ the component corresponding to state `TxFull,1`, which amounts to solving a simple linear equation system (see Appendix A). Sect. 4.3 shows results for probabilities and average frame sizes we obtained through measurements.

A similar model has been developed for the Send-And-Wait-ARQ protocol with k allowable trials. The major difference to the REPE model is the missing `TxPartial,i` states.

4.2 Experimental Setup

We implemented the REPE-ARQ scheme on the Tmote Sky platform in TinyOS 2.1 and integrated it with an existing CSMA MAC protocol. The MAC requires an idle channel before transmission and performs a maximum of three retransmissions in case an acknowledgement is not received correctly. We made measurements in the TWIST and MoteLab testbeds using the following setup: from the set of available nodes we selected one node to make periodic unicast transmissions (with a constant PPDU size of 133 byte) to a subset of about 25 other nodes in a round-robin fashion. In order to evaluate both, the REPE- and

[4] The model contains an approximation: in the calculation of c_F and c_P it is implicitly assumed that ACKs and NACKs are always received correctly.

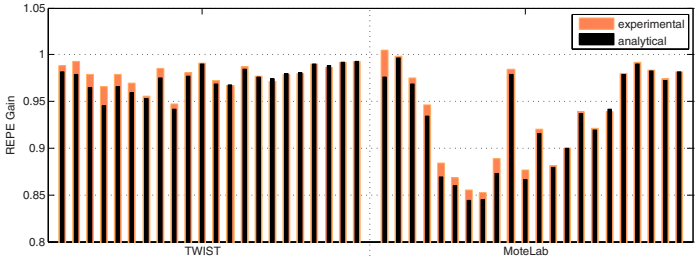


Fig. 13. The ratio of bytes transmitted with the REPE-ARQ vs. standard ARQ scheme (REPE Gain) during a 4-hour measurement / according to the analytical model. Each bar represents an 802.15.4 link and the bars are ordered by increasing PRR, ranging from about 0.1 to 0.99 (with a similar distribution as in Fig. 3).

the standard ARQ, under identical conditions, the REPE algorithm on the transmitter side was slightly modified: after the reception of a correct NACK the sender never reduced the frame size, but instead it inserted in the payload an additional, intermediate CRC at an offset where the reduced frame would have ended. In case of a bit error the receiver could thus determine whether the reduced-, the full-sized or both frames were affected[5], although in practice only a single full-sized frame was transmitted. With this approach both variants were running virtually *at the same time*. The sender/receiver could identify whether the transmission based on the REPE- or standard ARQ (or both) were (un)successful and count the number of transmitted bytes separately.

For example, assume that the first frame is completely destroyed; the second frame, i.e. first retransmission, is corrupted, the REPE-ARQ makes a correct estimate and the NACK (15 byte PPDU) is correctly received; the third frame has a single bit error, which is outside the region covered by the intermediate CRC and the corresponding ACK is received correctly; the fourth frame and its ACK are received correctly. Then the number of transmitted bytes accounted for the REPE-ARQ scheme is $133 + (133 + 15) + (40 + 11) + 0 = 332$ byte (assuming the retransmitted frame had a PPDU size of 40 byte and an ACK is 11 byte), for the standard ARQ it is $133 + 133 + 133 + (133 + 11) = 543$ byte.

4.3 Analytical and Experimental Results

Fig. 13 displays the experimental results from the TWIST and MoteLab testbeds. It shows a comparison of the REPE- with the standard ARQ scheme with respect to the total number of transmitted bytes per link during a 4-hour measurement (red bars). The results take all exchanged packets into consideration, including packets for which the REPE algorithm could not make a decision or packets that were inaccessible by software. From the traces we derived the probabilities $p_{F,Succ}$, $p_{P,Succ}$ and $p_{F,P}$, as well as the average frame sizes l_{NACK} and $l_{D,P}$

[5] Only in the first case the receiver would transmit an ACK frame and it would include a special flag to distinguish it from the ACK for a full-sized frame.

required by our analytical model (Sect. 4.1). The analytical results are shown as black bars in the same Fig. 13.

The experimental results differ for the two testbeds, possibly due to the diverse interference environments: in TWIST the gain is to 0.7 to 5.2 % (on average 2.1 %), while in MoteLab it is -0.5 to 14.7 % (on average 6.0 %). The highest gain is typically achieved on intermediate links with a PRR of around 0.6. On very good links the potential performance gain is low, because only a small fraction of the frames is corrupted and can be recovered in the first place. The analytical model matches these results quite well: the average difference is 0.0059 and the maximum is 0.029. This suggests that the model is suitable and that with a simple software extension on a typical mote platform the REPE scheme can indeed achieve a considerable performance gain.

5 Related Work

The REPE-ARQ scheme proposed in this paper is an instance of packet combining schemes [2]. In packet combining schemes a receiver stores erroneous packets to combine them with parts from later trials. A key issue is to identify the incorrect parts of a packet. In coded transmission systems it is possible that the decoder delivers not only decided bits, but also additional information specifying the confidence in its decisions on a per-bit basis (soft-information). In [14] soft-information is used in a selection-decode-and-forward relaying scheme. In the absence of true soft-information other methods are needed to infer the positions of the correct and incorrect parts of a packet. One method is to insert additional checksums into the packet [17].

The RSSI profiling technique discussed in this paper has not yet been proposed for usage in ARQ schemes, but RSSI has been used in other "non-traditional" contexts: Demirbas et al. [3] proposed to use the RSSI power-sum of acknowledgement frames in the process of estimating the number of neighbors for which a certain predicate is true. B-MAC [9] samples RSSI with high frequency to achieve a more accurate clear channel assessment; and in [10] per-bit RSSI information has been used (in conjunction with other per-bit/per-symbol information) to gather information about the cause of packet losses in Wifi systems and to adapt the MAC/PHY parameters accordingly: when a collision is inferred, the backoff process of the MAC is triggered whereas a weak signal results in an execution of the rate- or power-adaptation algorithm.

6 Conclusions and Future Work

In an experimental study we showed that by monitoring RSSI with high sampling rate during packet reception the receiver can often not only identify RF interference as the cause for packet corruption, but also estimate the bit error positions correctly. Our algorithm simply correlates the instantaneous signal strength with the arrival time of the individual symbols and marks all bits received while the signal level was above a certain threshold as incorrect. It is particularly effective

for large packets and when the interfering signal has short airtime. Through an analytical and experimental evaluation we showed that the approach can achieve high success rates and that an ARQ scheme generally benefits when coupled with our algorithm. We expect an additional performance gain if the radio hardware could provide more fine-grained signal strength information (shorter RSSI time windows or per-symbol LQI/correlation values).

Finding simple heuristics that can improve performance of the presented algorithm and investigating new strategies, such as identifying characteristic interferer "footprints" in the RSSI profile, seem a promising area of future work. Another interesting topic is the combination of REPE algorithms with coding schemes.

References

1. Bahl, P., Chandra, R., Moscibroda, T., Murty, R., Welsh, M.: White space networking with wi-fi like connectivity. In: SIGCOMM 2009: Proceedings of the ACM SIGCOMM 2009 conference on Data communication, pp. 27–38. ACM, New York (2009)
2. Dairaseh, A.-G.A., Baum, C.W.: Methods for packet combining in HARQ systems over bursty channels. MONET - Mobile Networks and Applications 2, 213–224 (1997)
3. Demirbas, M., Soysal, O., Hussain, M.: A singlehop collaborative feedback primitive for wireless sensor networks, April 2008, pp. 2047–2055 (2008)
4. Liu, H., Ma, H., El Zarki, M., Gupta, S.: Error control schemes for networks: an overview. Mob. Netw. Appl. 2(2), 167–182 (1997)
5. Abusubaih, M., Rathke, B., Wolisz, A.: Packet loss discrimination in multi-cell 802.11 wireless LANs. Technical Report TKN-08-010, Telecommunication Networks Group, Technische Universität Berlin (October 2008)
6. Maheshwari, R., Jain, S., Das, S.R.: A measurement study of interference modeling and scheduling in low-power wireless networks. In: SenSys 2008: Proceedings of the 6th ACM conference on Embedded network sensor systems, pp. 141–154. ACM, New York (2008)
7. Maheshwari, R., Jain, S., Das, S.R.: On estimating joint interference for concurrent packet transmissions in low power wireless networks. In: WiNTECH 2008: Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization, New York, NY, USA (2008)
8. Norris, J.R.: Markov Chains. Cambridge University Press, Cambridge (1997)
9. Polastre, J., Hill, J., Culler, D.: Versatile low power media access for wireless sensor networks. In: SenSys 2004: Proceedings of the 2nd international conference on Embedded networked sensor systems, Baltimore, MD, USA, pp. 95–107. ACM, New York (2004)
10. Rayanchu, S., Mishra, A., Agrawal, D., Saha, S., Banerjee, S.: Diagnosing wireless packet losses in 802.11: Separating collision from weak signal. In: IEEE INFOCOM 2008. The 27th Conference on Computer Communications, pp. 735–743 (2008)
11. Sentilla Corporation. Tmote sky datasheet, http://www.sentilla.com/moteiv-transition.html
12. Srinivasan, K., Levis, P.: RSSI is under appreciated. In: Proceedings of the Third Workshop on Embedded Networked Sensors (EmNets) (May 2006)

13. Texas Instruments. CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver (April 2002), http://focus.ti.com/docs/prod/folders/print/cc2420.html
14. Valentin, S., Volkhausen, T., Onat, F.A., Yanikomeroglu, H., Karl, H.: Enabling partial forwarding by decoding-based one and two-stage selective cooperation. In: Proc. IEEE Cognitive and Cooperative Wireless Networks (CoCoNet) co-located with IEEE ICC, Beijing, China (May 2008)
15. Handziski, V., Köpke, A., Willig, A., Wolisz, A.: TWIST: A scalable and reconfigurable testbed for wireless indoor experiments with sensor network. In: Proc. of the 2nd Intl. Workshop on Multi-hop Ad Hoc Networks: from Theory to Reality (RealMAN 2006), Florence, Italy (May 2006)
16. Werner-Allen, G., Swieskowski, P., Welsh, M.: MoteLab: a wireless sensor network testbed. In: IPSN 2005: Proceedings of the 4th international symposium on Information processing in sensor networks, Piscataway, NJ, USA, p. 68. IEEE Press, Los Alamitos (2005)
17. Willig, A.: Memory-efficient segment-based packet-combining schemes in face of deadlines. IEEE Transactions on Industrial Informatics 5(3), 338–350 (2009)

A Potentials of Markov Chains

Be $(X_n)_{n\geq 0}$ a time-homogeneous Markov chain with discrete (i.e. finite or countably infinite) state space $\mathcal{S}$ and state-transition matrix $\mathbf{P}$. The state space is partitioned into *inner states* D and *boundary states* or *final states* ∂D so that $\mathcal{S} = D \cup \partial D$. Suppose that $\mathbf{c} = (c_i)_{i\in D}$ and $\mathbf{f} = (f_i)_{i\in\partial D}$ are non-negative vectors representing the costs c_i when the chain is in the inner state $i \in D$ and the costs f_i when the chain is in the boundary state $i \in \partial D$. Let the random variable T be the hitting time for the boundary: $T = \inf\{n \geq 0 : X_n \in \partial D\}$. Set

$$\phi_i = \mathrm{E}_i\left[\sum_{n<T} c(X_n) + f(X_T)\cdot \mathbf{1}_{\{T<\infty\}}\right]$$

Then ϕ_i is the expected total costs when the chain starts in state $X_0 = i$ and operates in the inner states D, each time incurring a cost c_i, until it reaches a final state in ∂D, incurring a final cost corresponding to the final state. The final costs are incurred only when the hitting time T is finite. Then the following holds [8, Theorem 4.2.3]:

– The potential $\phi = (\phi_i)_{i\in\mathcal{S}}$ satisfies:

$$\begin{cases} \phi = \mathbf{P}\cdot\phi + \mathbf{c} & : \text{ in } D \\ \phi = \mathbf{f} & : \text{ in } \partial D \end{cases} \tag{1}$$

– If $\Pr_i[T<\infty] = 1$ (i.e. the probability to hit the final states when the starting state is $X_0 = i$) for all i then Equation 1 has at most one bounded solution.

In other words, we are looking for a solution of the system of linear equations given in 1.

F-LQE: A Fuzzy Link Quality Estimator for Wireless Sensor Networks*

Nouha Baccour[1,2], Anis Koubâa[2,3], Habib Youssef[4], Maissa Ben Jamâa[1], Denis do Rosário[2,5], Mário Alves[2], and Leandro B. Becker[5]

[1] ReDCAD Research Unit, National school of Engineers of Sfax, Sfax, Tunisia
[2] CISTER Research Unit, Polytechnic Institute of Porto (ISEP/IPP), Portugal
[3] Al-Imam Mohamed bin Saud University, Riyadh, Saudi Arabia
[4] Prince Research Unit, University of Sousse, Sousse, Tunisia
[5] Federal University of Santa Catarina, Brazil
nabr@isep.ipp.pt, aska@isep.ipp.pt, habib.youssef@fsm.rnu.tn, mbenj@redcad.org, dlr@isep.ipp.pt, mjf@isep.ipp.pt, lbecker@das.ufsc.br

Abstract. Radio Link Quality Estimation (LQE) is a fundamental building block for Wireless Sensor Networks, namely for a reliable deployment, resource management and routing. Existing LQEs (e.g. *PRR*, *ETX*, *Four-bit*, and *LQI*) are based on a single link property, thus leading to inaccurate estimation. In this paper, we propose *F-LQE*, that estimates link quality on the basis of four link quality properties: packet delivery, asymmetry, stability, and channel quality. Each of these properties is defined in linguistic terms, the natural language of Fuzzy Logic. The overall quality of the link is specified as a fuzzy rule whose evaluation returns the membership of the link in the fuzzy subset of good links. Values of the membership function are smoothed using EWMA filter to improve stability. An extensive experimental analysis shows that *F-LQE* outperforms existing estimators.

1 Introduction

In wireless sensor networks (WSNs), communication links are known to be extremely unreliable as they often experience significant quality fluctuations and weak connectivity. Link unreliability is partially due to the use of low-power radios, which are shown to be very sensitive to noise, interference, and multipath distortion.

Link quality estimation is a fundamental building block in the design of higher layer protocols, namely topology control, routing, and mobility management protocols. For instance, routing protocols rely on link quality estimation as a support mechanism to select the most stable routes for data delivery [1,2]. Stable routes are built by selecting links with the highest quality. Building such routes will improve the network throughput and maximize its lifetime, namely

* This work was funded by the ReDCAD reseach unit (05-UR-1403), by the CISTER Research Unit (FCT UI 608), and by EC FP7 EMMON and CONET projects.

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 240–255, 2010.

(*i.*) increasing the end-to-end probability of message delivery, (*ii.*) avoiding excessive re-transmissions over low quality links and (*iii.*) minimizing the route re-selection operation triggered by links failure.

Several link quality estimators (LQEs) have been reported in the literature [3,4,5,6,7]. They can be classified as either hardware-based or software-based. Existing LQEs (hardware or software) base their estimation on a single link property. However, other properties contribute to link quality, e.g. stability and channel quality. We alert the reader that we make a difference between channel quality and link quality. We define channel quality as a particular property of the communication link, which can be assessed by the *SNR* (Signal-to-Noise Ratio). Link quality represents the overall quality of the communication link, as it takes into account all (or a set of) link properties, including channel quality property.

In order to better estimate link quality, we advocate combining several important link properties, to get a holistic characterization of the link. In this paper, we propose a LQE that combines multiple metrics in order to achieve this goal. Link quality is affected by several aspects that are usually *imprecisely* measured. Fuzzy logic provides a convenient language to express and combine such imprecise knowledge. Thus in this work, we resort to fuzzy logic to estimate link quality. Individual link properties are stated in linguistic terms and combined in a fuzzy rule whose evaluation gives the degree of membership of the link in the fuzzy subset of good quality links.

The rest of this paper is organized as follows: in Section 2, we discuss the limitations of existing LQEs. In Section 3, we justify the use of Fuzzy Logic for link quality estimation. Then, we introduce our Fuzzy-link quality estimator (F-LQE) in section 4. Our experimental methodology for the performance evaluation of *F-LQE* is presented in Section 5 and experimental results are given in Section 6. We conclude in Section 7. We would like to mention here that the experimental results reported in section 6 confirm extensive simulation results obtained using TOSSIM. Details of the Simulation scenarios and results are omitted due to lack of space.

2 Limitation of Existing Link Quality Estimators

2.1 Hardware-Based Link Quality Estimators

Three LQEs belong to the family of hardware-based LQEs: *LQI* (Link Quality Indicator), *RSSI* (Received Signal Strength Indicator), and *SNR* (Signal-to-Noise Ratio). These estimators are directly read from the radio transceiver (e.g. the CC2420). Their advantage is that they do not require any additional computation. However, as reported in previous studies, hardware-based estimators do not provide accurate estimates [8,9,10,4], mainly for the following reasons: First, these metrics are measured based on the sample of the first 8 symbols of a received packet and not the whole packet. Second, these metrics are only measured for successfully received packets; therefore, when a radio link suffers from excessive packet losses, they may overestimate the link quality by not considering the information of lost packets. Third, despite the fact that hardware

metrics provide a fast and inexpensive way to classify links as either good or bad, they are incapable of providing a fine grain estimation of link quality [5].

The above limitations of hardware-based LQEs do not mean that this category of LQEs is useless. In fact, each of these LQEs (*SNR*, *LQI* and *RSSI*) provides a particular information on the link state, but none of them is able to provide a holistic information on the link quality. For instance, in [9], it has been reported that *RSSI* can provide a quick and accurate estimate of whether an incoming link is in or out of the grey area, whereas *LQI* can provide an estimate of where in the gray area a link is.

2.2 Software-Based Link Quality Estimators

Software-based LQEs enable to either count or approximate the reception ratio or the average number of packet transmissions/re-transmissions. Next, we recall some of the most widely adopted software-based LQEs.

The *PRR* counts the Packet Reception Ratio. It is computed as the ratio of the number of successfully received packets to the number of transmitted packets, for each window of w received packets. The Required Number of Packet retransmissions (RNP) [6] counts the average number of packet retransmissions required before a successful reception. It is computed as the ratio of the number of transmitted and retransmitted packets to the number of successfully received packets, minus 1 to exclude the first packet transmission. This metric is evaluated at the sender side for each w retransmitted packets.

The Window Mean with Exponentially Weighted Moving Average (WMEWMA) [3] and the Kalman filter based LQE [7] approximate the *PRR*. *WMEWMA* applies filtering on the *PRR* metric to smooth it, thus providing a metric that resists to transient fluctuation of *PRRs*, yet is responsive to major link quality changes. *WMEWMA* is then given by the following:

$$WMEWMA(\alpha, w) = \alpha \times WMEWMA + (1 - \alpha) \times PRR \tag{1}$$

where $\alpha \epsilon [0..1]$ controls the smoothness. This factor enables to give more importance, to the current *PRR* value (with $\alpha < 0.5$) or to the last *SPRR* value (with $\alpha > 0.5$). The Kalman filter based LQE [7] approximates the packet reception ratio based on *RSSI* and a pre-calibrated *PRR*/*SNR* curve.

On the other hand, the Expected Transmission Count (ETX) [11], and *four-Bit* [5] approximate the *RNP*. *ETX* is the inverse of the product of *PRR* of the forward link and the *PRR* of the backward link, which takes into account link asymmetry property. *Four-bit* is a sender-initiated estimator, already implemented in TinyOS. Like *ETX*, *four-bit* considers link asymmetry property. It combines two metrics (*i.*) $estETX_{up}$, as the quality of the unidirectional link from sender to receiver, and (*ii.*) $estETX_{down}$, as the quality of the unidirectional link from receiver to sender. The $estETX_{up}$ is exactly the *RNP* metric and $estETX_{down}$ approximates *RNP* as the inverse of *WMEWMA*, minus 1. The combination of the two metrics is performed through the EWMA filter as follow:

$$four\text{-}bit(w_a, w_b, \alpha) = \alpha \times four\text{-}bit + (1 - \alpha) \times estETX \tag{2}$$

estETX corresponds to $estETX_{up}$ or $estETX_{down}$: given w_a the beacon-driven estimation window and w_p the data-driven estimation window; at w_a received packets, the sender derives the *four-bit* estimate by replacing *estETX* for $estETX_{down}$ in Eq.2. At w_p transmitted/re-transmitted data packets, the sender derives the *four-bit* estimate by replacing *estETX* for $estETX_{up}$ in Eq.2.

Except of *four-bit*, these aforementioned LQEs rely on a single link quality metric, e.g. *PRR*, *SNR* or *RSSI*, to approximate either the reception ratio or the average number of packet transmissions/re-transmissions. However, as it has been shown [8,4,9], a single link quality metric assesses a particular link property and thus provides a partial characterization of the link. On the other hand, *four-bit* integrates two link quality metrics, namely *PRR* and *RNP*. However, it has the limitation of evaluating a single link aspect: the number of packet retransmissions, and does not take into account other important aspects, such as link stability level or channel quality. Further, *four-bit* combines two metrics having different nature, using the filter EWMA. Although filtering has been shown to be efficient to smooth the link quality estimates and provides a metric that resists to transient link quality changes [3], exploiting it for combining different metrics would lead to unstable link quality estimation [8].

3 Fuzzy Logic for Link Quality Estimation

The assessment of the quality of a wireless channel is a function of a number of metrics that are usually imprecisely estimated. Fuzzy logic provides a rigorous algebra for dealing with imprecise information. It is a mathematical discipline invented to express human reasoning in a rigorous mathematical notation. Unlike classical logic where a proposition is either true or false, fuzzy logic establishes the approximate truth value of a proposition based on linguistic variables and inference rules. Furthermore, fuzzy logic is a convenient method of combining conflicting objectives and expert human knowledge.

A linguistic variable is a variable whose values are words or sentences in natural or artificial language [12]. By using hedges like 'more', 'many', 'few', etc., and connectors like AND, OR, and NOT with linguistic variables, an expert can form rules, which will govern the approximate reasoning. In ordinary set theory, an element is either in a set or not in a set. In contrast, in fuzzy set theory, an element may partially belong to a set. A fuzzy set is defined as a class of objects with a continuum of grades of membership [13]. Formally, a fuzzy set A of a universe of discourse X = {x} is defined as A={x; μ_A(x) | $\forall$ x ϵ X}, where X is a space of points and μ_A(x) is a membership function of x ϵ X being an element of A. In general, the membership function μ_A(.) is a mapping from X to the interval [0,1]. If μ_A(x) = 1 or 0, $\forall$ x ϵ X, then the fuzzy set A becomes an ordinary set [13].

Example: Packet delivery is an important link property whose goodness is highly correlated with the overall goodness of the link. It can be evaluated by the *PRR* link quality metric. Let *PRR* be the Packet Reception Ratio across a given link. According to classical logic, a link is declared good when its *PRR* is

greater than a given threshold, say 0.95, and bad otherwise. For instance, given two different links, the first has a *PRR* equal to 95% and the second has a *PRR* equal to 94%. Classical logic declares only the first link as good. This example illustrates how *PRR* can only be imprecisely evaluated and classical reasoning fails to deal with such knowledge. Fuzzy Logic has been developed to handle this type of imprecise knowledge.

Let x ϵ [0..1] be a particular value of *PRR* and H be the fuzzy subset of links with high *PRR*. Then, for each x in the interval [0..1], $\mu_H(\mathrm{x})$ indicates the extent to which the link is considered having a high *PRR*, and $\mu_H(.)$ is the membership function of the fuzzy subset of links with high *PRR*. Packet delivery is considered as a fuzzy variable, which is expressed in linguistic terms such as low packet delivery and high packet delivery. The membership of the link in the Fuzzy set of high packet delivery links, is a matter of degree rather than a yes-no situation. It ranges in the interval [0..1]. By recalling the previous example, the first link with *PRR* equal to 95%, can have a degree of membership in the fuzzy subset of high delivery links, equal to 1, whereas the second link with *PRR* equal to 94%, can have a degree of membership of 0.9. A possible membership function of high packet delivery links is illustrated in Fig. 2 (refer to $\mu_{SPRR}(.)$).

During the lifetime of a WSN, the quality of a wireless channel is usually a function of several imprecisely measured channel properties, as packet delivery, asymmetry, and stability. Because of their imprecise nature, each such property can be conveniently expressed in linguistic terms. E.g., a channel can be unstable, stable, and highly stable. Each such term is a linguistic value for the linguistic variable channel stability. The numerical interpretation of each linguistic value is defined in the form of a fuzzy subset, characterized by a particular fuzzy membership function. Now, suppose that we want to combine multiple link properties to properly assess the link quality, each such combination is performed by a Fuzzy IF-THEN Rule. A fuzzy rule combines the linguistic variables using connectors (operators) such as AND and OR. The evaluation of the rule using a fuzzy operator (e.g. Yager operator [14]) returns a membership degree that represents the link quality estimate.

4 *F-LQE*: A Fuzzy Link Quality Estimator

4.1 Link Quality Metrics

In this section, we identify four link quality metrics to be considered in the design of *F-LQE*. Each metric describes an important link property. The set of selected link properties will be used in the next section to express the goodness of a given link.

Packet delivery is related to the capacity of the link to successfully deliver data. It is captured by some existing LQEs such as *PRR*, *WMEWMA*, and *ETX*, but not by others, such as *RNP*. *F-LQE* accounts for the packet delivery of the link by a measure of *SPRR*, which stands for Smoothed *PRR*. The *SPRR* is exactly the *WMEWMA* [3], described in section 2.

Asymmetry is the difference in connectivity between the uplink and the downlink. Communication between sensor nodes is usually bidirectional. Empirical studies such as [15] have shown that links asymmetry is due to the discrepancy in terms of hardware calibration, i.e. nodes do not have the same effective transmission power, reception sensitivity and noise floor. Therefore, it is not sufficient to estimate the link quality as the quality of the link in one direction . While some LQEs, such as *ETX* and *four-bit*, take into account link asymmetry, other estimators including *PRR*, *WMAWMA* and *RNP*, do not. *F-LQE* takes into account link asymmetry by measuring the difference between the uplink *PRR* (PRR_{up}) and the downlink *PRR* (PRR_{down}), noted as *ASL* (ASymmetry Level):

$$ASL(w) = |PRR_{up} - PRR_{down}| \tag{3}$$

The *ASL* metric gives an idea on whether a transmitted packet can be acknowledged or not. In fact, for a given sender, when the downlink is of high quality and the uplink is of bad quality, a correctly received packet would not be acknowledged or at least acknowledged after a certain number of retransmissions. The *ASL* captures this effect, which cannot be detected by the *PRR* alone.

Stability is the variability level of the link. Link stability is of a paramount importance for network protocols that preferably forward data over stable links in order to minimize retransmissions and topological changes. To the best of our knowledge, none of the existing LQEs takes into account this property. *F-LQE* assesses the stability of the link by the measure of the stability factor (SF), defined as the coefficient-of-variation of *PRR*. The *SF* metric is basically computed based on a history of 30 *PRRs*. We adopt the idea of "sliding window", for the update of the *PRRs* history at a new measure of *PRR*. We choose 30 as the history length to ensure a certain confidence for the computation of the coefficient of variation. Nevertheless, at network startup, we anticipate the computation of *SF* by considering only a history of 5 *PRRs* and as long as packets are received, the *PRRs* history is feeded back at every new measure of *PRR*, until collecting the 30 *PRRs* values.

Channel quality can be evaluated through the measure of the Signal-to Noise-Ratio (SNR). It has been shown in previous studies, such as [16] and [4] that although *SNR* alone is not able to give a holistic characterization of the link, it helps to enhance the accuracy of the link quality estimation. For example, a link that has a *PRR* near to 1 and a high *SNR*, e.g. 10 dBm (refer to Fig. 1), is significantly better than another link that has the same *PRR* but low *SNR*, e.g. 4 dBm, because the link quality of the second link is susceptible to drop considerably with a small change in the noise floor [4]. This observation can be clearly understood from Fig. 1.

The *SNR* metric can be derived by subtracting the noise floor (N) from the received signal (S), both in dBm. The S can be deduced by sampling the *RSSI* at the packet reception, and N can be derived from the *RSSI* sample just after the packet reception. In our proposed LQE, we average *SNR*, over w received packets to get *ASNR*: the link quality metric for the channel assessment.

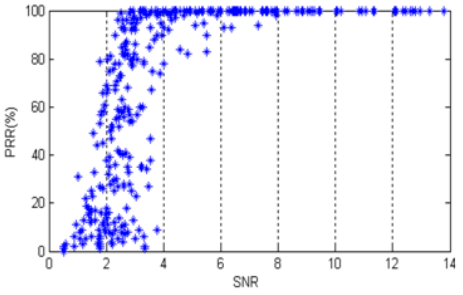


Fig. 1. *PRR/SNR*. For *ASNR* greater than 8dBm, the *PRR* is equal to 100%, and for *ASNR* less than 1 dBm, the *PRR* is less than 25%. In between, a small variation in the *ASNR* can cause a big difference in the *PRR*; links are typically in the transitional region.

4.2 Combination of Link Quality Metrics

F-LQE considers each of the link properties mentioned in the previous section as a different fuzzy variable. The goodness (i.e. high quality) of a link is characterized by the following rule:

IF the link has *high packet delivery* AND *low asymmetry* AND *high stability* AND *high channel quality* **THEN** it has *high quality*.

Here, *high packet delivery*, *low asymmetry*, *high stability*, *high channel quality*, and *high goodness* are linguistic values for the fuzzy variables packet delivery, asymmetry level, stability, channel quality, and quality (refers to link quality). Using and-like compensatory operator of [14], the above rule translates to the following equation of the fuzzy measure of the link i high quality.

$$\begin{aligned}\mu(i) = \beta.min(\mu_{SPRR}(i), \mu_{ASL}(i), \mu_{SF}(i), \mu_{ASNR}(i))+\\ (1-\beta).mean(\mu_{SPRR}(i), \mu_{ASL}(i), \mu_{SF}(i), \mu_{ASNR}(i))\end{aligned} \quad (4)$$

$\mu(i)$ is the membership in the fuzzy subset of high quality links. The parameter β is a constant in [0..1]. Recommended values for β are in the range [0.5..0.8] where 0.6 usually gives the best results [20], which is also confirmed in this work (see Section 6.1). μ_{SPRR}, μ_{ASL}, μ_{SF}, and μ_{ASNR} represent membership functions in the fuzzy subsets of high packet delivery, low asymmetry, low stability, and high channel quality, respectively. All membership functions have piecewise linear forms and then have low computation complexity. They are determined by two thresholds, as it is shown by Fig. 2.

The choice of the two thresholds, for the membership functions μ_{SPRR}, μ_{ASL}, and μ_{SF}, can be tuned according the application requirements. In our study, we have chosen reasonable values of these thresholds, with respect to each membership function. For instance, for μ_{SPRR}, for values of *SPRR* below 25%, the link is considered totally out of the fuzzy subset of links with high *PRR*. Starting from 95%, the membership to the fuzzy subset of links with high *PRR* is of 1.

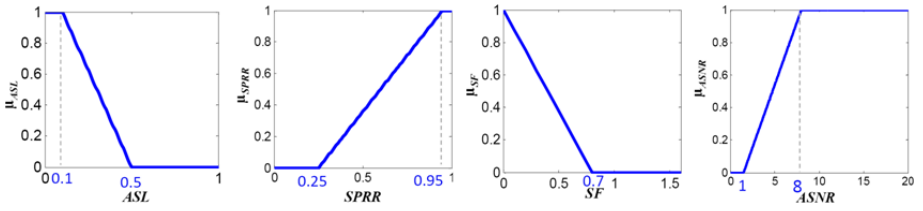


Fig. 2. Definition of membership functions μ_{SPRR}, μ_{ASL}, μ_{SF}, and μ_{ASNR}

For values of *SPRR* between 25% and 95%, the membership increases linearly from 0 to 1. The same reasoning holds for μ_{ASL}. The membership function μ_{SF} differs slightly to the other ones as the two thresholds are superposed. In fact, a link has 1 as membership to the fuzzy subset of links with high stability, only when the measured *SF* is equal to 1. Otherwise, its membership decreases linearly to achieves 0 when *SF* is equal to 0.7. The value 0.7 has been chosen by analyzing the *SF* of all experienced links.

The choice of the two thresholds for the membership function μ_{ASNR} depends on the environment and the hardware characteristics. Next, we present a detailed analysis for an efficient determination of these two thresholds.

In previous empirical studies, such as [4], based on the *PRR*/*ASNR* curve, the existence of two *ASNR* thresholds has been proven. When *ASNR* is larger than the first threshold, the *PRR* is greater than 95% almost all the time, which implies good channel. If *ASNR* is less than the second threshold, the *PRR* is lower than 25 % most of the time and the channel is bad. These thresholds are determined from the *PRR*/*ASNR* curve, which is in turn determined experimentally. In order to gather the *PRR*/*ASNR* curve, we carried out a set of experiments, using our testbed (refer to section V). Experiments were conducted under different network conditions (refer to TABLE 1). We generate the *PRR*/*ASNR* curve for each network setting. Fig. 1 depicts the *PRR*/*ASNR* curve for the default setting. The convenient choice of the two *ASNR* thresholds can be easily inferred from this curve. Notice that these thresholds are the same for all *PRR*/*ASNR* curves (settings), as we found that the curves have similar shapes.

The final step toward *F-LQE* computation is detailed in the rest of this section. We consider the following link quality metric (LQ):

$$LQ(w) = 100.\mu(i) \tag{5}$$

LQ combines *SPRR*, *ASL*, *SF* and *ASL* to provide a comprehensive assessment of the link. It attributes a score to the link, ranging in [0..100], where 100 is the best link quality and 0 is the worst. Using EWMA filter, we smooth *LQ* to get the *F-LQE* metric:

$$FLQE(\alpha, w) = \alpha.FLQE + (1 - \alpha).LQ \tag{6}$$

where, $\alpha = 0.9$, to provide stable link quality estimates. Notice that w is the estimation window, meaning that a node estimates link quality, i.e. computes *F-LQE*, based on each w received packets.

Eq.4 assumes that the mote has available data to compute the *SF* and the *ASL*. However, *SF* can be computed only when the mote has at least 5 measures of *PRR* and *ASL* can be computed only when the mote has both uplink and downlink *PRRs* (refer to Eq.3). Thereby, we introduced a simple mechanism that consists to the following: a node wishes to estimate link quality by considering different link properties, evaluated by the *SPRR*, *ASNR*, *ASL*, and *SF*. When one or both *ASL* and *SF* can not be computed due to the lack of some data, the node ignores the corresponding metric(s) in the computation of the membership function $\mu(i)$ in Eq. 4. For instance, when the node is not able to compute both *ASL* and *SF*, $\mu(i)$ in Eq. 4 becomes:

$$\mu(i) = \beta.min(\mu_{SPRR}(i), \mu_{ASNR}(i)) + (1-\beta).mean(\mu_{SPRR}(i), \mu_{ASNR}(i)) \quad (7)$$

5 Experimental Methodology

Our experimental study aims at analyzing and understanding the statistical properties of *F-LQE*, independently of any external factor, such as collisions and routing. These statistical properties impact its performance, in terms of *reliability* and *stability*. Reliability refers to the ability of the LQE to correctly characterize the link state. Stability refers to the ability to resist to transient (short-term) variations, also called fluctuations, in link quality. We compare the performance of *F-LQE* in terms of reliability and stability, with a set of well-known LQEs, namely *PRR*, *SPRR*, *ETX*, *RNP*, and *four-bit*.

Our testbed consists of a single-hop network with 49 TelosB motes [10], $N_1 \ldots N_{49}$, positioned in an outdoor environment (a garden at the university). The motes are distributed in a circular topology, as shown in Fig. 3. In this topology, 48 motes are divided in 8 sets with different radius. Each set contains 6 nodes, all placed in a circle around the central node N_1. The distance between two consecutive sets is equal to 0.75 meter. The first set, i.e. the nearest circle to N_1, has a radius of X meters, where X varies in $\{2, 3\}$. All TelosB motes are connected to a laptop PC using a combination of USB (Universal Serial Bus) cables and active USB hubs. We developed a software tool that runs on the PC to

Fig. 3. Nodes distribution according the circular topology, at an outdoor environment

Table 1. Experiment sets. Burst(X, Y, Z) and Synch(W, Y); X: Number of packets per burst, Y: inter-packets interval, Z: number of bursts, W: total number of packets.

	Traffic Type	**Packet Size**	**channel**
Impact of the Traffic Type	{Burst(100,100,10), Burst(200,500,4), Burst(100,1000,2), Synch(200,1000)}	28	26
Impact of the Packet Size	Burst(100,100,10)	{28, 114}	26
Impact of the Channel	Burst(100,100,10)	28	{20, 26}
Default Setting	Burst(100,100,10)	28	26

control and analyze the experiments. The control part, developed in Java, allows (*i.*) motes programming and control, (*ii.*) network configuration, and (*iii.*) data logging into a MySQL database. The data analysis (including graph generation) is performed in Matlab, allowing to work off-line. The motes are programmed in nesC [19] over TinyOS2.x environment.

In this study, we propose to estimate the quality of the unidirectional links $N_1 \longleftarrow N_i$. Since distance and direction are fundamental factors that affect link quality, we argue that by placing the nodes $N_2 \ldots N_{49}$ at different distances and directions from the central node N_1, the underlying links, $N_1 \longleftarrow N_i$, exhibit different qualities. Particularly, we choose a convenient X value so that most of the links are of intermediate qualities (belong to the transitional region) to better explore the performance of *F-LQE* as well as the other LQEs under evaluation.

After receiving the token, each couple of nodes (N_1, N_i), exchanges a certain number of data packets then passes the token to the next couple, (N_1, N_{i+1}). We considered two traffic patterns: *Bursty traffic* and *synchronized traffic*: For the Bursty traffic, N_1 sends a first burst of packets to N_i. When it finishes, it sends a notification to the PC, to allow N_i sending its burst of packets to N_1. When N_1 finishes sending, it notifies the PC. This operation is repeated for a certain number of bursts. As for the synchronized traffic, N_1 and N_i are synchronized to exchange packets (one packet a time). The PC sends a command to each mote to indicate the beginning of transmission time so that the mote sends its data in an exclusive time slot (to avoid collisions).

Based on exchanged data, the quality of links $N_1 \longleftarrow N_i$ has been estimated using *F-LQE*, as well as *PRR*, *SPRR*, *ETX*, *RNP*, and *four-bit*. We subject LQEs to different network conditions. In fact, we performed extensive experimentations through different experiments sets. In each experiments set we varied a certain parameter to study its impact, and for each parameter modification the experiment was repeated. Parameters under consideration were traffic type (3 sorts of burst and 1 synch), packet size (28/114), and channel (20/26). The duration of each experiment was approximately 8hs. TABLE 1 depicts the different settings for each experiments set. The transmission power was set to -25 dBm.

Like *F-LQE*, *four-bit* and *SPRR* use EWMA filter, which has an important parameter: the history control factor α. We chose $\alpha = 0.9$ for *four-bit*, as in [17], and $\alpha = 0.6$ for *SPRR*, as suggested in [3]. The estimation window w is a common parameter for all LQEs. In our study, we chose a small window, equal

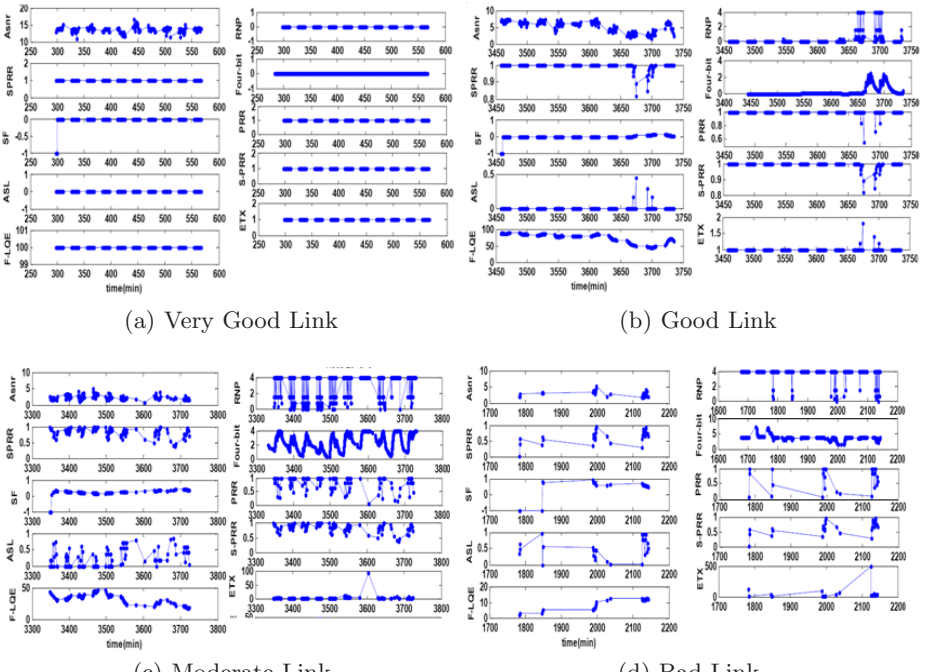
to 5 packets, for short-term link quality estimation. The same value of w is adopted in [17]. Further, in [18], it has been argued that short-time link quality estimation captures link dynamics at a high resolution in time.

6 Experimental Results

6.1 Reliability

The reliability of *F-LQE* is tested by studying (*i.*) the temporal behavior (Fig. 4), and (*ii.*) the distribution of link quality estimates, illustrated by the a scatter plot (Fig. 5) and the empirical cumulative distribution function, CDF, (Fig. 6).

Temporal Behavior: Fig. 4 uses four different links to show the temporal behaviour of each individual metric that constitutes *F-LQE* and its overall behavior. It also presents the results from other existing LQEs. From this figure, it can be observed that all LQEs agree that the first link (Fig. 4(a)) is of very good quality. This is expected since links of good quality are easy to estimate as they trend to be stable and symmetric [6,15]. On the other hand, moderate and bad links which are typically those of the transitional region and the disconnected region respectively, are more difficult to characterize.



(a) Very Good Link (b) Good Link

(c) Moderate Link (d) Bad Link

Fig. 4. Temporal behaviour of LQEs when faced with links with different qualities (Default Setting)

Fig. 4(b) shows how *F-LQE* outperforms existing LQEs because they are not able to distinguish between links, especially good links and very good links. In fact, let's observe the temporal behaviour of the link in Fig. 4(b), until the time *3660 min* (just before the link quality fluctuation). *PRR*, *SPRR*, and *ETX* are based on the *PRR* metric. They account for only one property : link delivery. These PRR-based LQEs declare the link as of very good quality. The same link quality state is declared by *RNP* and *four-bit*, which are RNP-based and accounts for a unique link property. However, our link should not have a very good quality due to the low *ASNR* value. In fact, the measured *ASNR* values are close to the receiver sensitivity. Consequently, the channel is of moderate quality, which prevents the link of being declared as "very good". In addition, the good properties that the link have are likely due to the constructive interference effect. On the other hand, *F-LQE* detects the real link state by considering different link properties. Indeed the link shown in Fig. 4(b) has some very good properties, including the delivery, the asymmetry and the stability, yet it has an *ASNR* of moderate quality which make of it a good link but not a very good link. From Fig. 4(c), we can observe how PRR-based LQEs, i.e. *PRR*, *ETX* and *SPRR* can overestimate link quality as they provide relatively high link quality estimates. The reason of this overestimation is the fact that *PRR*-based LQEs are only able to evaluate the link packet delivery property and they are not aware of the number of retransmissions to deliver a packet. A packet that is lost after one retransmission or after n retransmissions will produce the same estimate. On the other hand *RNP*-based LQEs, i.e.*RNP* and *four-bit*, can underestimate link quality by providing low link quality estimates. This underestimation is due to the fact that each of these LQEs assesses the required packet retransmissions and are not able to determine if these packets are received after these retransmissions or not. This discrepancy between *PRR*-based and *RNP*-based link quality estimates is justified by the fact that most of the packets transmitted over the link are correctly received (high *PRR*) but after a certain number of retransmissions (high *RNP*). More importantly, each of these LQEs assess a single and different link property. *F-LQE* estimates the link not as good as *PRR*-based estimators do, and not as bad as *RNP*-based estimators do. It takes into account different properties to provide a holistic characterization of the real link state.

Fig. 4(d) gives a preliminary idea on the stability of *F-LQE* as well as the other LQEs (a detailed analysis of the stability of *F-LQE* is given in section 6.2). Indeed, the link shown in Fig. 4(d) is generally of bad quality. Furthermore, this link is a *bursty* link, as its quality can turn to good (e.g. *PRR* equal to 1 and *RNP* equal to 0), yet in the short term. *F-LQE* is a stable LQE as it resists to these short-term link quality fluctuation whereas the other LQEs are not stable as their link quality estimates switch temporarily to very good estimates.

Now, let us see more arguments for *F-LQE* reliability by analyzing the distribution of link quality estimates.

Link Quality estimates distribution: Form the scatter plot of Fig. 5, we can see that *F-LQE* estimates are more scattered than those of the other link

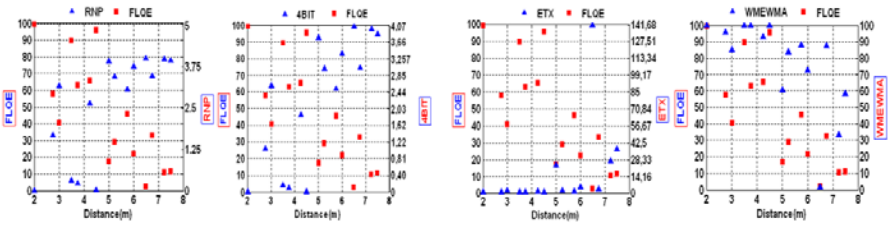


Fig. 5. Scatter plot of each LQE according to distance (Default Setting)

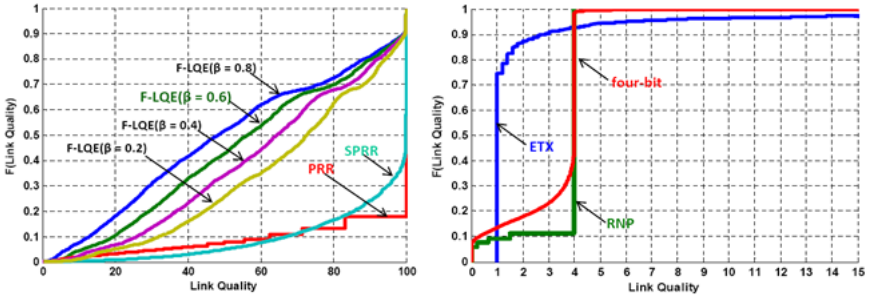
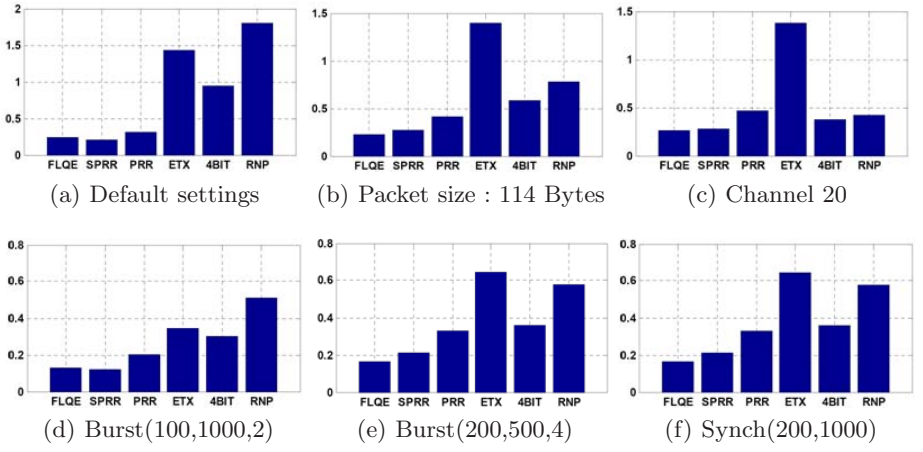


Fig. 6. Empirical CDFs of LQEs (Default Setting)

estimators. For example, the *RNP* estimates are mostly aggregated to 4 retransmissions (the maximum). That means that two links assumed to have different qualities, may be aggregated to have almost the same qualities when using *RNP* as LQE; and they would have different qualities when using *F-LQE* as LQE. The same thing holds for the rest of LQEs. This observation shows that *F-LQE* would surely perform better than the existing LQEs. Hence again, we show the reliability of *F-LQE* as it is able to provide a fine grain classification of links.

The above observations can be confirmed if we look into the CDF plot in Fig. 6. This plot is obtained based on all the links and the default setting (refer to Table 1). Notice that we did not include the CDF plots for the other settings, as they have similar shape as the CDF plot based on the default setting. Fig. 6 shows that *PRR*, *SPRR* and *ETX* overestimate link quality as they estimate most of the links to have good quality. In contrast, *RNP*, and *four-bit* underestimate link quality as they consider most of the links having bad quality. In between, *F-LQE* provides reasonable link quality estimates (neither overestimate nor underestimate link quality). Furthermore, the distribution of link quality estimates is nearly an uniform distribution, which means that *F-LQE* is able to to distinguish between links having different link qualities. These observations confirm the reliability of *F-LQE*.

In our study, we have set β (refer to Eq.4) to 0.6. In the following, we justify this choice by studying the impact of β on the reliability of *F-LQE*. Fig. 6



(a) Default settings (b) Packet size : 114 Bytes (c) Channel 20

(d) Burst(100,1000,2) (e) Burst(200,500,4) (f) Synch(200,1000)

Fig. 7. Sensitivity to transient fluctuation in link quality, for different network settings

shows the effect of β on the CDF. From this figure, we retain two important findings: First, the higher β is, the more pessimistic *F-LQE* is. This is completely reasonable, since by increasing β, we give more importance to the *min* (refer to Eq.4). Second and more importantly, by choosing β equal to 0.6, we get the nearest distribution to the uniform distribution, which justify the choice of β.

6.2 Stability

A link may show transient link quality fluctuations due to many factors principally related to the environment, and also to the nature of low-power radios, which have been shown very prone to noise. LQEs should resist to these fluctuations and provide stable link quality estimates. This property is of paramount importance in WSNs. For instance, routing protocols do not have to reroute information when a link quality show transient degradation, because rerouting is a very energy and time consuming operation.

To reason about this issue, we measure the sensitivity of the LQEs to transient fluctuations by the coefficient of variation of its estimates. Fig. 7 compares the sensitivity (stability) of *F-LQE* with that of *PRR*, *ETX*, *SPRR*, *RNP*, and *four-bit*, with respect to different setting (refer to Table 1). According this figure, we retain two observations: First, generally, *F-LQE* is the most stable LQE. Second, except *ETX*, *PRR*-based LQEs, i.e. *PRR* and *SPRR*, are more stable than *RNP*-based LQEs, i.e. *RNP* and four-bit. *ETX* is PRR-based, yet it is shown as unstable. The reason is that when the *PRR* tends to 0 (very bad link) the *ETX* will tend to infinity, which increase the standard deviation of *ETX* link estimates.

7 Conclusion

In this paper, we have presented a novel link quality estimator (F-LQE) for wireless sensor networks (WSNs). In contrast to existing LQEs, which only assess one single link property thus providing a partial view on the link, *F-LQE* combines four link metrics (*SPRR*, *ASNR*, *ASL*, and *SF*) using Fuzzy Logic, since we believed (and proved) to be an appropriate strategy to fuse different and imprecise metrics. The overall quality of the link is then specified as a Fuzzy IF-THEN rule, which combines the four metrics, viewed as linguistic variables. The evaluation of the fuzzy rule returns the membership of the link in the fuzzy subset of good links. *F-LQE* has been evaluated extensively both by simulation and experimentation, demonstrating greater performance over existing solutions, in terms of reliability and stability. The simulations were conducted using TOSSIM. Details of the Simulation scenarios and results are omitted due to lack of space.

Future work will address the impact of *F-LQE* on higher layer protocols (e.g. routing) and its use as a basic building block for proposing time-efficient mobility management mechanisms in WSNs. We also envisage to turn *F-LQE* implementation in TinyOS available to the community as an open-source.

References

1. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P.: Collection Tree Protocol. To appear in Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, SenSys (2009)
2. Woo, A., Tong, T., Culler, D.: Taming the underlying challenges of reliable multihop routing in sensor networks. In: Proceedings of the 1st international conference on Embedded networked sensor systems, SenSys 2003 (2003)
3. Woo, A., Culler, D.: Evaluation of efficient link reliability estimators for low-power wireless networks. EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-03-1270 (2003), http://www.eecs.berkeley.edu/Pubs/TechRpts/2003/6239.html
4. Lal, D., Manjeshwar, A., Herrmann, F., Uysal-Biyikoglu, E., Keshavarzian, A.: Measurement and characterization of link quality metrics in energy constrained wireless sensor networks. In: IEEE Global Telecommunications Conference (2003)
5. Fonseca, R., Gnawali, O., Jamieson, K., Levis, P.: Four bit wireless link estimation. In: Proceedings of the Sixth Workshop on Hot Topics in Networks (2007)
6. Cerpa, A., Wong, J.L., Potkonjak, M., Estrin, D.: Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing (2005)
7. Senel, M., Chintalapudi, K., Lal, D., Keshavarzian, A., Coyle, E.J.: A kalman filter based link quality estimation scheme for wireless sensor networks. In: IEEE Global Telecommunications Conference (2007)
8. Baccour, N., Koubaa, A., Ben Jamaa, M., Youssef, H., Zuniga, M., Alves, M.: A comparative simulation study of link quality estimators in wireless sensor networks. To appear in the 17th IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (2009)

9. Srinivasan, K., Levis, P.: Rssi is under appreciated. In: Proceedings of the Third Workshop on Embedded Networked Sensors (2006)
10. 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 (2005)
11. Couto, D.S.J.D., Aguayo, D., Bicket, J., Morris, R.: A highthroughput path metric for multi-hop wireless routing. In: Proceedings of the 9th annual international conference on Mobile computing and networking (2003)
12. Zadeh, L.A.: The concept of a linguistic variable and its application to approximate reasoning. Information Sciences 8, 199–249 (1975)
13. Zadeh, L.A.: Fuzzy sets. Information and Control 8, 338–353 (1965)
14. Yager, R.R.: On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Trans. Syst. Man Cybern. 18(1), 183–190 (1988)
15. Zhao, J., Govindan, R.: Understanding packet delivery performance in dense wireless sensor networks. In: Proceedings of the 1st international conference on Embedded networked sensor systems (2003)
16. Yunqian, M.: Improving wireless link delivery ratio classification with packet snr. In: International Conference on Electro Information Technology (2005)
17. Fonseca, R., Gnawali, O., Jamieson, K., Levis, P.: Four Bit Wireless Link Estimation. In: Proceedings of the Sixth Workshop on Hot Topics in Networks (2007)
18. Becher, A., Landsiedel, O., Wehrle, K.: Towards short-term wireless link quality estimation. In: Hot Emnets (2008)
19. Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., Culler, D.: The nesc language: A holistic approach to networked embedded systems. In: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation (2003)
20. Youssef, H., Sait, S.M., Khan, S.A.: Fuzzy Evolutionary Hybrid Metaheuristic for Network Topology Design. In: Proceedings of the International Conference on Evolutionary Multi-criteria Optimization (2001)

On the Mechanisms and Effects of Calibrating RSSI Measurements for 802.15.4 Radios

Yin Chen and Andreas Terzis

Computer Science Department
Johns Hopkins University
{yinchen,terzis}@jhu.edu

Abstract. Wireless sensor network protocols and applications, including those used for localization, topology control, link scheduling, and link quality estimation, make extensive use of Received Signal Strength Indication (RSSI) measurements. In this paper we show that inaccuracies in the RSSI values reported by widely used 802.15.4 radios, such as the CC2420 and the AT86RF230, have profound impact on these protocols and applications. Furthermore, we experimentally derive the response curves which translate actual RSSI values to the raw RSSI readings that the radios report and show that they contain non-linear and even non-injective regions. Fortunately, these curves are consistent across radios of the same model, making RSSI calibration practical. We present a calibration mechanism that removes the artifacts in the raw RSSI measurements, including ambiguities created by the non-injective regions in the response curves, and generates calibrated RSSI readings that are linear. This calibration removes many of the outliers generated when raw RSSI readings are used to estimate Signal to Noise (and Interference) ratios, estimate radio model parameters, and perform RF-based localization.

1 Introduction

The IEEE 802.15.4 standard specifies that a radio's PHY layer must provide an 8-bit integer value as an estimate of the received signal power [9]. This value is commonly known as the Received Signal Strength Indication (RSSI) in the wireless sensor networks (WSN) community. Numerous WSN protocols use RSSI measurements extensively, including those for localization [8,12,22,24], link quality estimation [13,19], packet reception ratio modeling [23] and transmission power control [11,17,18].

While many protocols directly use the RSSI measurements that the radios provide, the standard only requires that the reported RSSI values should be linear and within ±6 dB of the actual RSSI values. However, ±6 dB is a wide error margin. For example, Packet Reception Ratio (PRR) can decrease from 100% to 0% with a 2 or 3 dB difference in the received signal strength [13]. The consequence of this observation is that possible inaccuracies in the reported RSSI values can profoundly impact applications that rely on RSSI measurements.

In this paper we examine two 802.15.4 compliant radios, the widely used Chipcon/TI CC2420 [20] and Atmel AT86RF230 [2], and show that they do indeed

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 256–271, 2010.

introduce systematic errors in the RSSI measurements they provide. As a matter of fact, the coarse *RSSI value vs. input power* graph included in the CC2420 datasheet hints at the existence of non-linearities. Nevertheless, the manufacturer states that the RSSI response curve is very linear [20]. We independently derive high resolution RSSI response curves using a variable signal generator and verify the existence of the non-linearities hinted by the CC2420 datasheet. We also note that the AT86RF230 datasheet does not provide an equivalent graph. Fortunately, these response curves are radio-specific but device independent. In other words, different physical devices that use the same model of radio have identical response curves. Consequently, mitigating these nonlinearities does not require calibrating each device individually.

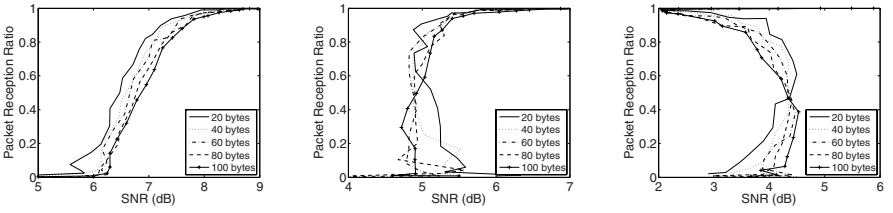
This result allows us to develop a generic calibration scheme to compensate for the radio's inaccuracies. Specifically, we derive a reference RSSI response curve which determines the calibrated RSSI value for the raw RSSI value that the radio reports. However, due to the existence of the non-injective regions in which a raw RSSI value maps to multiple actual RSSI values, the reference RSSI response curve is not able to always provide the necessary mapping. To resolve this problem, we leverage the ability of 802.15.4 radios to transmit at multiple power levels and dynamically fit a receiver's raw RSSI measurements to the radio's RSSI response curve. This approach provides an excellent fit and is able to accurately resolve the ambiguities that non-injective regions generate. Finally, we present the profound impact of the RSSI nonlinearities and quantify the benefits of the proposed calibration scheme on a wide variety of applications.

The paper has five additional sections. The section that follows reviews background material on low-power radios and RSSI measurements. Section 3 presents the results of an experiment that investigates the influence of packet size on the Packet Reception Ratio (PRR). Section 3 also presents the mechanism we developed to derive the high resolution RSSI response curves exhibiting the nonlinearities mentioned previously. Section 4 details the proposed RSSI calibration scheme, while Section 5 presents the adverse impact of the RSSI nonlinearities on various protocols and applications, and the improvements that the proposed calibration scheme achieves. We close in Section 6 with a brief discussion.

2 Background

Many of the popular hardware platforms in wireless sensor networks today use radios complying to the IEEE 802.15.4 standard [9]. This standard was developed specifically for low-power and low-cost embedded devices and implementations from multiple vendors are available today.

One such implementation, the TI/Chipcon CC2420 [20] is used in multiple platforms [5,15]. It allows the user to select one of eight output power levels, ranging from -25 dBm to 0 dBm. The Atmel AT86RF230 [2] is another 802.15.4 radio, used in the Iris mote [6]. In addition to higher receiver sensitivity this radio transmits at one of 16 power levels, from -17.2 dBm to 3 dBm.



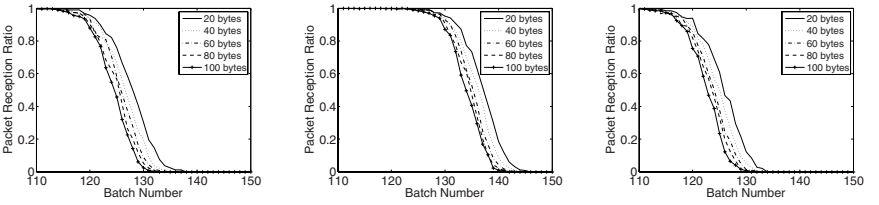
(a) Mote 1. (b) Mote 2. (c) Mote 3.

Fig. 1. Packet Reception Ratios (PRR) as a function of Signal to Noise ratio (SNR). The curves were experimentally derived from three different motes using CC2420 radios. PRR curves for multiple payload sizes are generated for each case. While the results from cases (a) and (b) follow the expected pattern, the pattern in case (c) is counter-intuitive, with PRR improving as SNR decreases.

Both radios provide an 8-bit register which indicates the strength of the received radio signal (RSSI). The 8-bit RSSI value is averaged over 8 radio symbol periods, i.e., 128 μs. Reported RSSI values are measured in dBm, in one dBm increments. There are two categories of RSSI measurements. The first category measures the strength of the radio signal corresponding to a received packet, while the second measures the power of the ambient channel noise. Using these two RSSI values, one can compute the Signal-to-Noise ratio (SNR) for a received packet. We will refer to these two types of RSSI values as **signal RSSI** and **noise RSSI** respectively throughout the rest of this paper. Furthermore, we name the RSSI values provided by the radio chips as **raw RSSI** or **reported RSSI** interchangeably. We will show that reported RSSI values are nonlinear with respect to actual received signal power, defined as **actual RSSI**. The calibration scheme introduced in Section 4 can eliminate the nonlinearity and we term the resultant RSSI values as **calibrated RSSI**.

As part of our effort to improve the fidelity of the TOSSIM simulator [10], we performed an experiment, detailed in Section 3.1, to generate a model for the relationship between Packet Reception Ratio (PRR) and SNR. Specifically, TOSSIM does not consider the packet's size when determining whether it will be successfully received. This simplification can underestimate or overestimate the packet loss that an application will experience in practice because packet sizes can vary from a few bytes (e.g., ACKs) to above one hundred bytes.

Figure 1 presents the PRR versus SNR curves we experimentally derived using three Tmote Sky motes and various payload sizes. SNR values are determined using the reported RSSI values. It is evident from this figure that there is no consistent correlation between packet size and PRR. More alarmingly, Figure 1(c) suggests that PRR improves as SNR decreases! We will show in the next section that this counter-intuitive (and incorrect) result is the consequence of the RSSI nonlinearity.



(a) Mote 1. (b) Mote 2. (c) Mote 3.

Fig. 2. PRR versus batch number for three motes with various payload sizes. All mote curves are consistent, shifted only by X-offsets corresponding to location differences.

3 Accuracy of RSSI Measurements

3.1 Influence of Packet Size on Packet Reception Ratio

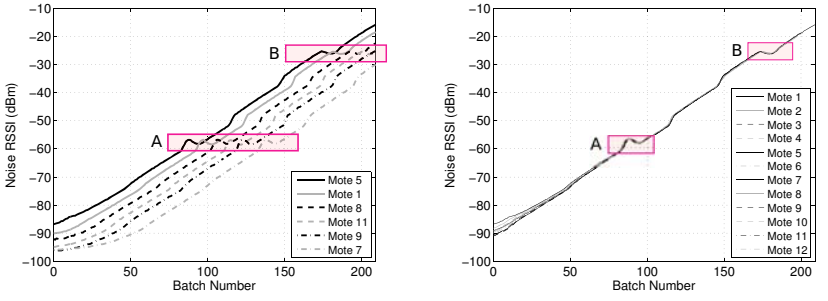
We conducted the packet size experiment in an indoor testbed comprising 13 Tmote Sky motes equipped with CC2420 radios [20]. The motes were placed at fixed locations in a quiet office and were powered through their USB ports to eliminate variations due to different battery power levels. A sole transmitter periodically broadcasted packets to the other motes. Furthermore, to minimize interference from co-located WiFi networks, we used 802.15.4 channel 26 that does not overlap with any 802.11 b/g channels [14].

Considering that the mote locations are fixed and radios can transmit at only eight power levels[1], we generate a wide range of SNR values by varying the ambient noise level N. We do so by generating noise signals of variable power levels using a Universal Software Radio Peripheral (USRP) [7]. The noise signal the USRP generates has an almost flat power spectral density within the frequency range of 802.15.4 channel 26.

In this experiment, we increase the noise strength linearly (in dBm) using a constant step size. The linearity was validated using the Anritsu MS2721B spectrum analyzer [1]. At each noise strength level, the transmitter broadcasts a batch of 2,500 packets of five different payload sizes. To minimize the impact of temporal variations in the radio channel, the transmitter broadcasts packets with different sizes at an inter-packet interval of 25 ms. For each batch of received packets we calculate the PRR and average SNR using reported RSSI at each receiver mote. Figure 1 presents the results of these calculations for three receiver motes. It is clear from the mote-specific patterns that different motes report different results. The results in Figure 1(c) are especially puzzling, suggesting that hardware variations or even faults may be at play.

We use Figure 2, which plots the PRR versus batch number curves generated from the same data, to verify that the radios function correctly. Note that noise strength increases with each successive batch, while the signal strength remains

[1] While the CC2420 datasheet mentions a total of 31 possible transmission levels, it specifies the output power levels for only eight of them.



(a) 6 motes, unaligned. (b) 12 motes, aligned.

Fig. 3. (a) RSSI measurements reported by six Tmote Sky motes as the noise strength increases linearly in dBm. While the response is mostly linear, it includes multiple nonlinear regions. Similar results from six additional motes are omitted in the interest of clarity. (b) Aligned RSSI response curves for all twelve Tmote Sky motes. Device-specific variations are minimal. Boxes A and B indicate the non-injective regions.

constant. The SNR therefore decays as batch number increases and thus one expects that PRR will decrease accordingly. Indeed, Figure 2 confirms this trend. Furthermore, unlike Figure 1, the results from the three motes are consistent. The X-axis offsets are due to the different locations of the motes, leading to different received signal strengths and noise levels. This result indicates that the underlying cause of the discrepancies shown in Figure 1 is not device variability or failure. Instead we posit that they are due to inaccuracies in the RSSI values that the motes report, leading to inaccurate SNR calculations.

3.2 RSSI Response Curves

Next, we design an experiment to derive high resolution RSSI response curves and verify the hypothesis in the previous section that the inaccurate reported RSSI values lead to the results presented in Figure 1.

We conducted this experiment in the same indoor testbed used for the previous experiment. However, unlike the previous experiment, there is no mote transmitter. Instead, twelve Tmote Sky motes periodically sense the noise signal that the USRP generates. The benefit of this approach is that it allows us to generate signals with a much wider range of transmit powers, compared to the eight levels available from the CC2420. Like the previous experiment, the strength of the USRP noise increases linearly (in dBm) with each successive batch, therefore the actual RSSI at the motes should also be linear with respect to batch number. We note that although the radios report integer RSSI values, sub-dBm accuracy can be achieved by averaging a series of RSSI measurements. We also note that the noise strength increment per batch is different from the previous experiment.

Figure 3(a) illustrates the RSSI measurements recorded by six of the twelve motes. We omit the results from the remaining motes in the interest of clarity

because they show similar patterns. It is evident that the noise RSSI curve for every mote can be divided into several major linear segments. Within each segment, the mapping between the noise RSSI and the batch number is linear. Moreover, the slopes for all the linear segments are almost equal. In the transitional regions that connect these linear segments, however, the mapping between the noise RSSI and the batch number is linear with different slopes or even nonlinear. Furthermore, some of these transitional regions are not monotonically increasing. This violates the most important assumption about RSSI: RSSI readings should be higher for stronger signals. In fact, this assumption is the basis of range-free localization mechanisms [8]. Also, due to the existence of these non-monotonic regions, the mapping from actual signal strength to the RSSI readings that the radios report is non-injective. Considering that the nonlinearities exist for all the motes tested, we categorize them as systematic errors in the RSSI measurements by the CC2420 radio.

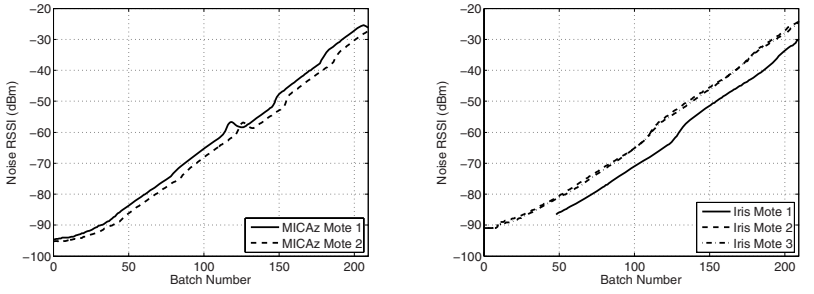
Another important observation from Figure 3(a) is that the mote-specific RSSI curves are considerably similar. In fact, the major difference among the curves is the offset on the X-axis. This is mainly due to the different signal strength attenuations resulting from the varying distances between individual motes and the USRP. Given this similarity, we select one RSSI curve as the reference and align the other curves to it. Figure 3(b) shows the result of this process. It is clear that overall the RSSI curves for different motes match very well. The mismatches at the lower end of the graph are likely due to the fact that RSSI readings in this region are approaching the ambient noise level.

We note that the results in Figure 3(b) were achieved by shifting the RSSI curves only along the X-axis. This is desirable because it suggests that even though nonlinear and non-injective regions exist, they occur at the same reported RSSI values for different motes. In other words, the device-specific variations regarding the nonlinearity and non-injectiveness are minimal. Consequently, mitigating these errors does not require calibrating each device individually.

3.3 Platform and Radio Variability

In order to investigate the influence of the hardware platform on RSSI measurements, we performed the same experiment using two MICAz motes. MICAz motes use the same CC2420 radio but are otherwise different from the Tmote Sky motes used thus far. Figure 4(a) presents the RSSI response curves for two MICAz motes. It is clear that the curves in Figure 4(a) are very similar to the ones in Figure 3. This similarity indicates that the RSSI measurement errors are caused by the CC2420 radio chip itself and are platform independent.

Finally, to investigate whether the observed nonlinearities are specific to the CC2420 radio, we performed the same experiment using three Crossbow Iris motes which use the AT86RF230 radio. Figure 4(b) presents the results from this experiment. While different from those in Figures 3 and 4(a), the RSSI response graphs of the IRIS motes exhibit consistent nonlinearities. On the other hand, the RSSI response graphs do not exhibit non-injective regions. Finally, we observe



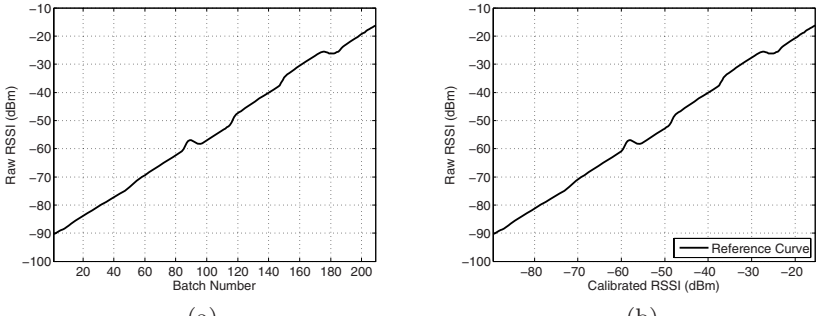
(a) MICAz motes. (b) IRIS motes.

Fig. 4. (a) RSSI response curves from two MICAz motes using the same CC2420 radio as the Tmote Sky. Response curves are consistent across platforms that use the same radio. (b) RSSI response curves for three IRIS motes using the AT86RF230 radio. While the radio responds differently from the CC2420 radio, it also has nonlinear regions.

consistent non-linearities across all three motes, indicating that the systematic errors in AT86RF230 raw RSSI measurements are also device independent.

4 RSSI Calibration

The results from the previous section show that radios have a non-linear, yet consistent response curve that maps the actual received signal strength to reported RSSI measurements. Figure 5(a), derived from combining the individual curves in Figure 3(b), shows such a response curve for the CC2420 radio. The issue with this curve is that the X-axis is in units of batch number instead of actual RSSI values. The noise strength increases linearly with respect to the batch number and therefore the relationship between batch number (n) and actual RSSI (r) should be $r = \alpha \times n + \beta$. The noise strength increment α can be



(a) (b)

Fig. 5. (a) Combination of the 12 curves in Figure 3(b). (b) The reference RSSI curve for CC2420 radios, derived by linearly transforming the X-axis from (a).

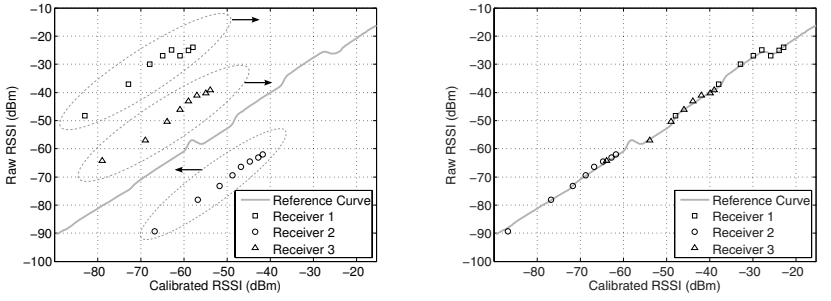


Fig. 6. (a) Aligning a set of eight pairs $(P_i - L, R_i)$ to the reference curve for three motes, with $x = P_i - L$ and $y = R_i$. The relative positions among the eight points for each mote are fixed. (b) All mote measurements align well to the reference curve.

measured experimentally, using the Anritsu MS2721B. On the other hand, measuring β accurately would require measuring the power of the signal that comes out of the receiving mote's antenna with a pre-calibrated receiver. Fortunately, as we explain next, we do not need to estimate β accurately.

Any errors in estimating β will lead to a constant offset between the actual and *calibrated* RSSI. However, this offset is not important because it does not affect the SNR and SINR calculations. Furthermore, because the offset is consistent across different devices that are using the same model of radio, directly comparing calibrated RSSI values is equivalent to comparing actual RSSI values. Considering these arguments, we settle for estimating calibrated RSSI $r' = \alpha \times n + \beta' = r + \epsilon$ and we select β' such that batch number $n = 140$ corresponds to calibrated RSSI $r' = -40$ dBm. We selected the (140,-40) pair because it makes the reported RSSI values almost equal to the calibrated RSSI values in most of the linear regions of the curve. Figure 5(b) presents the result of this translation.

Figure 5(b) can then be used to translate raw RSSI readings to calibrated RSSI values. This figure however cannot resolve the ambiguities in the non-injective regions, in which a raw RSSI value maps to multiple calibrated RSSI values. Fortunately, we can leverage the ability to control the transmitter's power to resolve these ambiguities as we describe next.

Consider the case in which the raw RSSI value R_1 for a received packet lies within one of the non-injective regions of Figure 5(b). The receiver then requests the transmitter to reveal the power level P_1 used to transmit that packet and to transmit additional packets using different power levels $P_2, \ldots, P_m$[2]. The receiver records the raw RSSI values $R_2, \ldots, R_m$ for each of the additional packets. If at least one of the R_i's falls within the radio's injective response region, it is possible to translate it to the calibrated RSSI value R'_i via Figure 5(b). Note

[2] The number m is upper-bounded by the number of available transmit power levels from the radio, and the actual P_i values are listed on the radio's datasheet.

that $R'_i = P_i - L$, where L is the link attenuation in dB. Knowing the values for both P_i and R'_i we can solve for L. Then $P_1 - L$ can be assigned to be the calibrated RSSI value corresponding to R_1, because L is consistent across different transmit powers. The computational cost is trivial, because only one raw RSSI value (R_i) needs to be translated into the calibrated RSSI (R'_i), via a lookup table corresponding to the reference curve.

To be more robust against measurement errors and noise, we can also select the value of L that minimizes the mean square difference between the m points $(P_1 - L, R_1), \ldots, (P_m - L, R_m)$ and the reference curve. The computational cost would be increased in this case because multiple table lookups are necessary.

Figures 6(a) and 6(b) present an example of the m-point calibration process for three receivers, with $m = 8$, equal to the number of power levels available in CC2420. One can see that the eight points for each mote fit well to the reference curve. Note that generally m can be arbitrarily chosen between 2 and the number of available power levels.

5 Applications

In what follows we explore the impact of RSSI calibration in modeling, protocol behavior, application performance, and simulation veracity.

5.1 PRR-SNR Model

First, we investigate the benefits of applying the RSSI calibration mechanism described in Section 4 to the problem of understanding the relationship between PRR and SNR. In turn, this understanding can be used in a variety of applications ranging from online link estimation to link modeling and simulation.

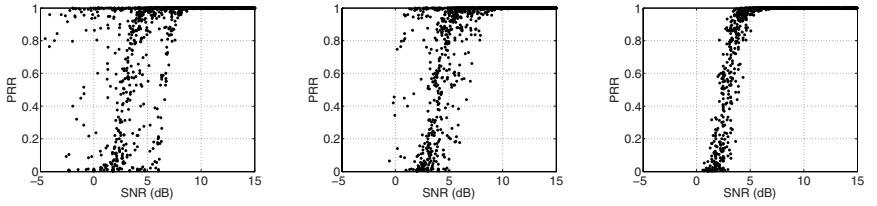
We conducted this experiment in the same indoor testbed used for the packet size experiment. One Tmote Sky mote was chosen as the transmitter while the other twelve motes acted as receivers. However, unlike the packet size experiment, all packets had the same size. Moreover, the transmitter varied the output power levels to produce a larger range of SNR values.

The signal to noise ratio (SNR) is computed as $SNR = \frac{S}{N}$, where S is the power of the received packet and N is the power of the ambient noise. Let both S and N be measured in milliwatts (mW). In logarithmic scale the above equation becomes $SNR_{dB} = S_{dBm} - N_{dBm}$ where S_{dBm} and N_{dBm} are the logarithmic scale powers of the received signal and ambient noise respectively.

In order to measure S and N, the receivers record both packet RSSI (S_{RSSI}) and noise RSSI (N_{RSSI}). Then, $S_{RSSI} = 10 \log_{10}(S+N)$ and $N_{RSSI} = 10 \log_{10} N$. Therefore, S_{RSSI} is essentially the sum of the power of the radio signal and the power of the noise. Nevertheless, when $S_{RSSI} \gg N_{RSSI}$ one can approximate SNR as $SNR_{dB} \approx S_{RSSI} - N_{RSSI}$. On the other hand, when S_{RSSI} is comparable to N_{RSSI} we need to compute SNR through

$$SNR_{dB} = 10 \log_{10}(10^{S_{RSSI}/10} - 10^{N_{RSSI}/10}) - N_{RSSI} \quad (1)$$

because $S = 10^{S_{RSSI}/10} - 10^{N_{RSSI}/10}$.



(a) SNR calculated using Equation 1.

(b) SNR calculated using Equation 2.

(c) SNR calculated using the RSSI calibration scheme.

Fig. 7. Experimentally derived PRR vs. SNR curves, calculated using three increasingly accurate schemes. Using calibrated instead of raw RSSI measurements can remove most of the outliers in the PRR vs. SNR relationship.

We use Equation 1 to calculate the SNR values used in the PRR vs. SNR scatter plot shown in Figure 7(a). One can see from this figure that there is a large transitional region, through which the relationship between PRR and SNR is noisy and unpredictable. The existence of this transitional region has been widely reported in the wireless sensor networks literature [21,23,25].

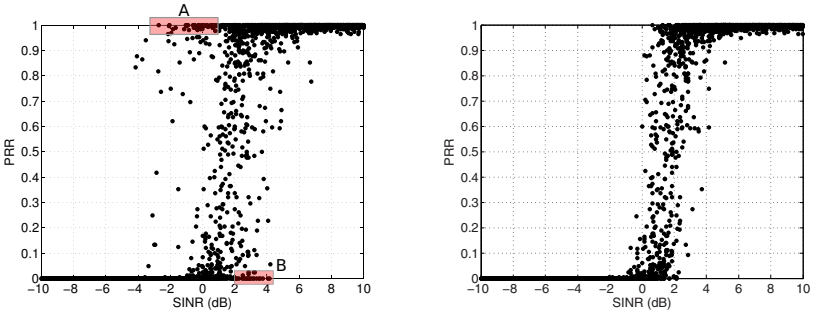
At the same time, given the nonlinearity presented in Figure 3(b), using raw RSSI values to calculate SNR can be problematic. For instance, if S and N are both within the non-injective regions, the reported RSSI value for their sum might be smaller than the reported RSSI value for S or N alone.

To eliminate this issue, we configured the transmitter to broadcast one additional batch of packets at each of the eight output power levels, while keeping the USRP turned off. This allows us to use the packet RSSI measurements directly, without having to calculate S from S_{RSSI} and N_{RSSI}. In this case, we denote the reported packet RSSI value as $\hat{S}_{RSSI}$, and calculate SNR as

$$SNR_{dB} = \hat{S}_{RSSI} - N_{RSSI} \tag{2}$$

Doing so assumes that the channel conditions do not change dramatically throughout the course of the experiment. This is however reasonable, as the measurements were collected at night when the environment at our indoor testbed was static. We use Equation 2 to calculate the SNR values used in Figure 7(b). One can see that the extent of the transitional region is considerably smaller compared to Figure 7(a). This observation validates our intuition that Equation 1 is polluted by the nonlinearities in the measurement of S_{RSSI}. At the same time, the SNR in Equation 2 is computed using the raw values for $\hat{S}_{RSSI}$ and N_{RSSI} and therefore it is also susceptible to the nonlinearities' adverse effects.

Finally, Figure 7(c) shows the equivalent scatter plot when the SNR in Equation 2 was calculated using the calibrated RSSI values for $\hat{S}_{RSSI}$ and N_{RSSI}. It is evident from Figure 7(c) that the transitional region becomes significantly smaller compared to the previous two graphs. This result indicates that the RSSI nonlinearity can account for a large portion of the noise and outliers in the PRR vs. SNR model.



(a) Raw SINR. (b) Calibrated SINR.

Fig. 8. PRR vs. SINR results for 2 concurrent transmitters and 14 receivers. The calibrated SINR shown in (b) eliminates most of the outliers present at the raw SINR graph shown in (a). Links in boxes A and B are the extreme outliers that complicate the SINR-PRR modeling.

5.2 SINR Modeling and Concurrent Transmission

The previous section investigated the relationship between PRR and SNR. When multiple transmitters are active at the same time, they start to interfere with each other and PRR is determined by another metric, the SINR (Signal to Interference and Noise Ratio). Maheshwari et al. conducted an extensive study on the relationship between SINR and PRR for the CC2420 radio [13]. However, the SINR-PRR graphs in [13] have a remarkable volume of outliers for which high SINR links have low PRR, while links with negative SINR exhibit high PRR. Maheshwari et al. thus concluded that the SINR-PRR model is still far from perfect to be employed in TDMA scheduling [13].

We conjecture that the CC2420 RSSI nonlinearity accounts for some of the outliers seen in [13]. In order to validate this conjecture, we performed an experiment with two Tmote Sky motes configured to broadcast simultaneously to 14 Tmote Sky motes. Figure 8(a) and 8(b) present the derived uncalibrated and calibrated SINR-PRR scatter plots. One can clearly see that in our experiment, most of the outliers were indeed introduced by the CC2420 RSSI nonlinearity. Approximately 25% of the links in this experiment experience more than 2 dB change in their SINR values when applying the calibration scheme. For the data points located within the $[-4, 5]$ dB region in Figure 8(a), 8% are outliers. After calibration, 94% of these outliers are corrected in Figure 8(b).

Accurate SINR models are important to protocols, such as CMAC [17], that attempt to schedule multiple, non-interfering transmissions. Specifically, CMAC utilizes an SINR-PRR model to set the nodes' transmission powers such that multiple interfering links can be used concurrently. Doing so can significantly increase system throughput. Nevertheless, the outliers exposed in Figure 8(a) can lead CMAC to suboptimal transmission schedules.

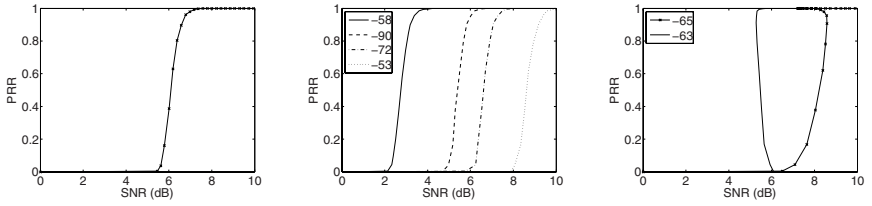


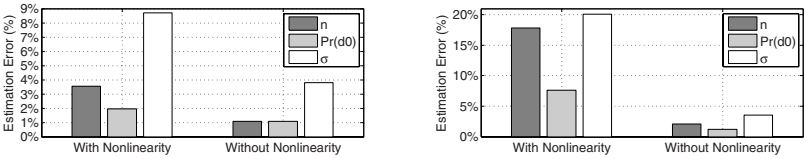
Fig. 9. PRR vs. SNR curves generated by a TOSSIM simulation. (a) shows the curve from the original TOSSIM. Curves in (b) and (c) are derived from a modified version of TOSSIM that simulates the CC2420 RSSI nonlinearity. Each curve in (b) and (c) was derived by keeping the noise power constant and varying signal strength to create a dynamic SNR range. The noise power listed in the legends is in dBm units.

For example, we observed in the experiment that one of the two senders (mote 0) could deliver $> 98\%$ of its packets to receiver 11 when transmitting at -7 dBm, while the other sender (mote 1) could deliver at the same time $> 98\%$ of its packets to receiver 14 using transmit power of -15 dBm. However, the raw SINR value calculated at mote 14 is -0.128 dB which translates to a very low PRR according to the SINR-PRR model. For this reason, a power scheduling protocol based on the SINR model, such as CMAC [17], would not schedule mote 1 to transmit at power -15 dBm. On the other hand, if CMAC used the calibrated SINR value at mote 14 (= 2.2056 dB) it would correctly schedule the concurrent transmission. We note that link $1 \rightarrow 14$ is one of the links in box A shown in Figure 8(a).

5.3 WSN Simulation

Existing wireless sensor network simulators such as TOSSIM [10] do not simulate the radio-specific RSSI measurement nonlinearities. Nevertheless, it is straightforward to integrate RSSI response curves, such as the one in Figure 5(b), to these simulators. Doing so requires constructing a lookup table and using linear interpolation to convert actual RSSI values (i.e., X-axis in Figure 5(b)) into reported RSSI values (i.e., Y-axis in Figure 5(b)).

We implemented such a mechanism for TOSSIM and Figure 9 presents a few sample PRR-SNR curves. Specifically, Figure 9(a) shows the PRR versus reported SNR curve in the current version of TOSSIM. Without the integration of the RSSI response curve, the shape of this PRR-SNR curve does not change as RSSI varies. In contrast, Figures 9(b) and 9(c) show that different curves emerge as we vary the power of the ambient noise, due to the nonlinearity in the reported RSSI values. In particular, the curves in Figure 9(c) resemble the experimentally derived curves in Figures 1(b) and 1(c).



(a) $n = 3.5$, $P_r(d_0) = -30$, $\sigma = 3$ (b) $n = 2$, $P_r(d_0) = -30$, $\sigma = 3$

Fig. 10. Errors in estimating log-normal path loss model parameters

5.4 Estimating Radio Propagation Model Parameters

A variety of WSN applications and protocols rely on radio propagation models. The first step in using such a model is to estimate the corresponding model parameters. This step is usually accomplished by deploying motes to record the radio signal strength (i.e., RSSI), at various locations within the area of interest. Therefore, the non-linearities of RSSI measurements can directly pollute the estimation of the model's parameters and thus the performance of the protocols that rely on the model's accuracy.

A commonly used radio propagation model is the log-distance path loss model with log-normal shadowing [16]. According to this model, the received signal strength $P_r(d)$ (in dBm) at a given distance d from the transmitter is given by:

$$P_r(d)[dBm] = P_r(d_0)[dBm] - 10n \log(\frac{d}{d_0}) - X_\sigma \tag{3}$$

where $P_r(d_0)$ is the expected signal strength at reference distance d_0, n is the path-loss exponent, and $X_\sigma \sim N(0, \sigma)$ is a normal random variable (in dB).

In order to investigate the impact of CC2420 RSSI nonlinearity on parameter estimation, we simulate the procedure of deploying motes at various distances from a transmitter to derive the log-normal parameters $P_r(d_0)$, n and σ. Specifically, we generate the $P_r(d)$ samples using a set of log-normal parameters and use the RSSI measurements to estimate those parameters. We note that doing so assumes that the log-normal model perfectly characterizes the RF propagation, a premise which might be violated in reality. Nevertheless, this treatment isolates the sources of errors in model parameter estimation and therefore allow us to focus on the errors that the RSSI nonlinearity introduces. A total of 240 samples were generated, corresponding to measurements collected at locations uniformly spaced at distances between 1 and 30 meters from the transmitter. Two samples were generated for each distance. Figure 10 presents the estimation errors with and without the presence of the CC2420 RSSI nonlinearity for two sets of model parameters. It is clear that the nonlinearity can cause significant errors. Errors in estimating these parameters can directly impact the applications that rely on them, such as RF based localization [24] and network coverage prediction [4].

5.5 RF Based Localization

Localization techniques based on RF signal strength use RSSI measurements to estimate the distances of a mobile device to several reference servers whose locations are known. *Trilateration* can then be used to estimate the device's location [24]. The previous section demonstrated that the nonlinearities in the CC2420 RSSI measurements impact the estimation of the radio model parameters. In turn, these errors can directly diminish the accuracy of such localization algorithms. On the other hand, localization schemes that employ RSSI signatures should intuitively be less affected by such nonlinearities. For example, the RADAR system collects a database of RSSI signatures by having a mobile node broadcast packets to three reference servers from a set of known locations [3]. The resulting RSSI measurements collected at the three servers, along with the mobile device's location, form the 5-tuples $[RSSI_1, RSSI_2, RSSI_3, X, Y]$ that constitute the localization database.

Once this training phase is complete, a device that needs to estimate its location broadcasts a series of packets to the reference servers. The system then finds the entry in the localization database with the minimum mean square difference from the RSSI measurements and uses the entry's $[X, Y]$ coordinates as the estimate of the node's current location. The MoteTrack system extends this simple approach and makes it highly robust and decentralized [12].

We performed an experiment similar to the one performed for RADAR in a 20 m^2 room using four Tmote Sky motes. Three of the motes were setup as reference servers while the fourth played the role of the mobile device. A total of 70 locations were tested and the mobile device was configured to broadcast at the seven transmission powers at 25 ms intervals.[3] Thus seven databases were constructed corresponding to the seven power levels. Each database was then used to evaluate localization errors for the corresponding transmission power. The method we used to estimate localization accuracy is the same with the one used by the original RADAR mechanism [3]. Namely, we select one of the database entries and try to localize it using only the other database entries. The localization error is then equal to the Euclidean distance between the entry's actual location and the location of the closest database entry. We iterate through all the database entries in this way and calculate the average localization error.

Table 1 lists the resulting localization errors. It is evident from the table that different transmission powers lead to different errors. This should not happen if the RSSI readings were linear, because a linear constant does not change the mean square difference, the metric used to select the most similar record from the database[4]. The table's last row indicates that calibrating the raw RSSI measurements reduces the localization error.

[3] The lowest transmission power (-25dBm) was not sufficient to ensure packet reception at the reference servers from all the tested locations.

[4] Assuming that the signal strength is significantly higher than the ambient noise, which was true during the course of this experiment.

Table 1. Localization errors for the RSSI-signature-based localization technique as a function of transmission power. The rightmost column represents the localization error as a percentage on top of the error achieved using the calibrated RSSI measurements.

Transmission Power (dBm)	Average Localization Error (cm)	Percentage
-15	138.97	7%
-10	133.19	2%
-7	148.80	14%
-5	146.79	13%
-3	137.39	5%
-1	134.10	3%
0	140.26	8%
Calibrated	130.35	-

6 Conclusion

This paper verifies the existence of the oft-ignored RSSI non-linearities for the popular Chipcon/TI CC2420 802.15.4 radio and shows that similar non-linearities exist in the Atmel AT86RF230 radio. Furthermore, the paper experimentally derives the non-linear RSSI response curves for the two radios, shows that they are consistent across devices that use the same model of radio, and proposes a scheme to calibrate raw RSSI measurements including those that fall within a curve's non-injective regions. Last but not least, we evaluate the impact of non-linearities in RSSI measurements on PRR modeling, WSN simulation, as well as protocols for concurrent link scheduling and RF-based localization.

The implications of our results to future designs are twofold. First, protocol and application designers need to be mindful that RSSI response curves may be non-linear or even non-injective and include techniques to compensate for such non-linearities. Second, considering the dependence of multiple protocols on RSSI measurements, future radio designs should strive to produce linear or at least injective RSSI response curves.

Acknowledgments

We extend our gratitude to Neal Patwari, Phil Levis, and Prabal Dutta for their insightful comments and suggestions. We would also like to thank the anonymous reviewers that for helping us improve the paper's presentation. This research was supported in part by NSF grants CNS-0834470 and CNS-0546648. Any opinions, finding, conclusions or recommendations expressed in this publication are those of the authors and do not represent the policy or position of the NSF.

References

1. Anritsu Company. Spectrum Master MS2721B
2. Atmel Corporation. AT86RF230: Low Power 2.4 GHz Transceiver for ZigBee, IEEE 802.15.4, 6LoWPAN, RF4CE and ISM applications
3. Bahl, P., Padmanabhan, V.N.: RADAR: An In-Building RF-based User Location and Tracking System. In: Proceedings of INFOCOM (2000)

4. Chipara, O., Hackmann, G., Lu, C., Smart, W.D., Roman, G.-C.: Radio mapping for indoor environments. Technical report, Washington University in St. Louis (2007)
5. Crossbow Corporation. MICAz Specifications (2004)
6. Crossbow Corporation. Iris Specifications (2007)
7. Ettus Research LLC. Universal Software Radio Peripheral (2007)
8. He, T., Huang, C., Blum, B.M., Stankovic, J.A., Abdelzaher, T.: Range-free localization schemes for large scale sensor networks. In: MobiCom 2003, pp. 81–95. ACM, New York (2003)
9. IEEE Standard 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs) (May 2003)
10. Levis, P., Lee, N., Woo, A., Welsh, M., Culler, D.: TOSSIM: Accurate and scalable simulation of entire TinyOS Applications. In: Proceedings of Sensys 2003 (November 2003)
11. Lin, S., Zhang, J., Zhou, G., Gu, L., Stankovic, J.A., He, T.: ATPC Adaptive Transmission Power Control for Wireless Sensor Networks. In: Proceedings of the 4^{th} ACM Sensys Conference (2006)
12. Lorincz, K., Welsh, M.: Motetrack: a robust, decentralized approach to rf-based location tracking. Personal Ubiquitous Comput 11(6), 489–503 (2007)
13. Maheshwari, R., Jain, S., Das, S.R.: A measurement study of interference modeling and scheduling in low-power wireless networks. In: Proceedings of Sensys 2008, pp. 141–154. ACM, New York (2008)
14. Musaloiu-E, R., Terzis, A.: Minimising the effect of wifi interference in 802.15.4 wireless sensor networks. Int. J. Sen. Netw. 3(1), 43–54 (2007)
15. Polastre, J., Szewczyk, R., Culler, D.: Telos: Enabling Ultra-Low Power Wireless Research. In: IPSN/SPOTS 2005 (April 2005)
16. Rappaport, T.S.: Wireless Communications: Principles & Practices. Prentice Hall, Englewood Cliffs (1996)
17. Sha, M., Xing, G., Zhou, G., Liu, S., Wang, X.: C-MAC: Model-driven Concurrent Medium Access Control for Wireless Sensor Networks. In: Proceedings of IEEE Infocom (2009)
18. Son, D., Krishnamachari, B., Heidemann, J.: Experimental study of concurrent transmission in wireless sensor networks. In: Proceedings of ACM Sensys (2006)
19. Srinivasan, K., Levis, P.: RSSI is Under Appreciated. In: Proceedings of the 3rd Workshop on Embedded Networked Sensors (EmNets) (May 2006)
20. Texas Instruments. CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver (2006)
21. Woo, A., Tong, T., Culler, D.: Taming the underlying challenges in reliable multihop wireless sensor networks. In: Proceedings of ACM Sensys (2003)
22. Yedavalli, K., Krishnamachari, B., Ravula, S., Srinivasan, B.: Ecolocation: a sequence based technique for rf localization in wireless sensor networks. In: Proceedings of IPSN 2005, Piscataway, NJ, USA, p. 38. IEEE Press, Los Alamitos (2005)
23. Zamalloa, M.Z., Krishnamachari, B.: An analysis of unreliability and asymmetry in low-power wireless links. ACM Transactions on Sensor Networks (June 2007)
24. Zanca, G., Zorzi, F., Zanella, A., Zorzi, M.: Experimental comparison of rssi-based localization algorithms for indoor wireless sensor networks. In: REALWSN 2008, pp. 1–5. ACM, New York (2008)
25. Zhao, J., Govindan, R.: Understanding Packet Delivery Performance In Dense Wireless Sensor Networks. In: Proceedings of the ACM Sensys (November 2003)

Making Sensornet MAC Protocols Robust against Interference

Carlo Alberto Boano[1], Thiemo Voigt[2], Nicolas Tsiftes[2], Luca Mottola[2], Kay Römer[1], and Marco Antonio Zúñiga[3]

[1] Institut für Technische Informatik, Universität zu Lübeck, Lübeck, Germany
[2] Swedish Institute of Computer Science (SICS), Kista, Sweden
[3] Digital Enterprise Research Institute (DERI), Galway, Ireland

Abstract. Radio interference may lead to packet losses, thus negatively affecting the performance of sensornet applications. In this paper, we experimentally assess the impact of external interference on state-of-the-art sensornet MAC protocols. Our experiments illustrate that specific features of existing protocols, e.g., hand-shaking schemes preceding the actual data transmission, play a critical role in this setting. We leverage these results by identifying mechanisms to improve the robustness of existing MAC protocols under interference. These mechanisms include the use of multiple hand-shaking attempts coupled with packet trains and suitable congestion backoff schemes to better tolerate interference. We embed these mechanisms within an existing X-MAC implementation and show that they considerably improve the packet delivery rate while keeping the power consumption at a moderate level.

1 Introduction

The increasing number of wireless devices sharing the same unlicensed ISM bands affects both reliability and robustness of sensornet communications. Sensor networks that operate, for example, in the 2.4 GHz band must compete with the communications of WLAN, Bluetooth, WirelessUSB, and other 802.15.4 devices. They may also suffer the interference caused by appliances such as microwave ovens, video-capture devices, car alarms, or baby monitors. Such problems will increase when more of these devices will be deployed in the near future.

Interference may have a deteriorating effect on communication, as it leads to packet loss and lack of connectivity. This may result in worse performance and reduced energy efficiency of sensornets, causing major issues in a number of application domains, e.g. safety-critical applications in industry and health care.

Studying the impact of interference has been hard because of the lack of proper tools that enable an inexpensive generation of controlled interference. Recently, we demonstrated a method to generate customized and repeatable interference patterns using a common CC2420 radio transceiver in special mode [1]. Using that method, we experimentally study the impact of interference on several MAC protocols, such as Contiki's NULLMAC, X-MAC, LPP, and CoReDac; and

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 272–288, 2010.

TinyOS's LPL. Our goal is to find effective mechanisms that handle interference properly. We carry out our experiments in the 2.4 GHz ISM band, which is also the most crowded one.

In this paper, we investigate which mechanisms improve the robustness of communication in congested networks while remaining reasonably energy efficient. In our experiments we identify three methods that can increase the robustness of sensornet MAC protocols against interference. Since low-power MAC protocols allow nodes to turn off their radio most of the time, they require some kind of *handshaking*. For example, in X-MAC a receiver needs to hear a strobe and answer with a strobe acknowledgment [2]. In Low Power Probing (LPP), the opposite happens: a sender waits for a probe from the intended receiver before it can send the packet [3]. Our experiments show that protocols or parameter settings that enable potentially more handshakes in case some fail due to interference are more robust. Another method that we identify is to use *packet trains* that enable the sender to quickly send multiple packets that have been accumulated during an interference period. The third method is the selection of suitable *congestion backoff schemes* when using Clear Channel Assessment (CCA) and detecting a busy channel. Based on these findings, we include these mechanisms in an X-MAC version, and show its improved robustness to interference.

Our contributions are the following. First, to the best of our knowledge, we are the first to experimentally study how interference affects different MAC protocols. Second, we identify mechanisms that enable MAC protocols to sustain high packet delivery rates while using low-power consumption even in presence of interference. Third, we show experimentally that the choice of congestion backoff schemes is critical for communication performance and energy efficiency in congested networks. Fourth, we augment an existing X-MAC implementation with these mechanisms, and demonstrate substantial performance improvements.

Our paper proceeds as follows. Section 2 provides an overview on the investigated MAC protocols. We describe the methodology and the setup of our experiments in Section 3. Thereafter, in Section 4 and 5, we present our experimental results and identify methods that handle interference properly. In Section 6 we design a new version of X-MAC that implements several of the identified methods and evaluate its performance. We review related work in Section 7 and present our conclusions in Section 8.

2 Background

Medium access control for wireless sensor networks has been a very active research area for the past couple of years, and the literature provides an amazing number of different implementations and incremental improvements. In our work, we exploit the four MAC layers available in Contiki (NULLMAC, X-MAC, LPP, CoReDac) and Tiny OS' LPL. Section 2.1 briefly describes these protocols, and Section 2.2 explains the role of CCA in sensornet MAC protocols.

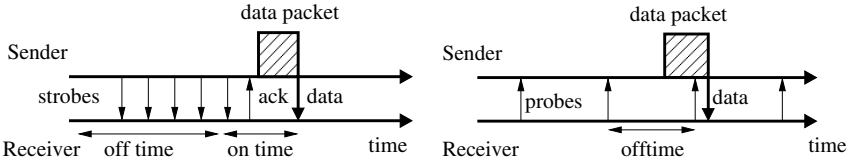


Fig. 1. In X-MAC (left), the sender strobes until the receiver is awake and can receive a packet. In LPP (right), the receivers send probes to announce they are awake and ready to receive packets.

2.1 Overview of Used MAC Protocols

NULLMAC. NULLMAC is a minimalistic MAC protocol that simply forwards traffic between the network layer and the radio driver. As such, it does not provide any power-saving mechanism, and keeps the radio always on. This allows for the maximum throughput achievable, while consuming the highest amount of energy. When used with CCA and back-off timers, NULLMAC behaves as a traditional CSMA-CA protocol. Because of these characteristics, we use NULLMAC as a baseline to compare the performance of other protocols, and to verify the correctness of our setup.

X-MAC. X-MAC is a power-saving MAC protocol [2] in which senders use a sequence of short preambles (strobes) to wake up receivers. Nodes turn off the radio for most of the time to reduce idle listening. They wake up shortly at regular intervals to listen for strobes. When a receiving node wakes up and receives a strobe destined to it, it replies with an acknowledgment indicating that it is awake. After receiving the ACK, the sender transmits the data packet, as shown in in Figure 1(a).

The X-MAC implementation in Contiki has several parameters of significance to our experiments. *Ontime* determines the maximum time that a receiver listens for strobes, whereas *offtime* specifies the time to sleep between waking up to listen for strobes. *Strobe_time* denotes the duration a sender transmits strobes until it receives a strobe acknowledgment from the receiver. In the default Contiki X-MAC implementation, strobe_time = offtime + (20 × ontime).

Low-Power Probing (LPP). LPP is a power-saving MAC protocol where receivers periodically send small packets, so called probes, to announce that they are awake and ready to receive a data packet [3]. After sending a probe, the receiver keeps its radio on for a short time to listen for data packets. A node willing to send a packet turns on its radio waiting for a probe from a neighbor it wants to send to. On the reception of a probe from a potential receiver, the node sends an acknowledgment before the data packet, as shown in Figure 1(b).

The LPP implementation in Contiki contains two important parameters. *Ontime* determines how long a receiver keeps the radio on after the transmission of

a probe, *offtime* is the time between probes. We use $\frac{1}{2}$ and $\frac{1}{64}$ seconds for *offtime* and *ontime* respectively. Another parameter is the time to keep an unsent packet: Contiki LPP's default value is $4\times$(ontime + offtime). If LPP receives a packet from the network layer when the packet queue is full, LPP discards the new packet. The queue length is configurable, and the default size is 8 packets.

Low-Power Listening (LPL). We consider a Low-Power Listening (LPL) layer that implements an asynchronous wake-up scheme for CC2420 radios [4]. Nodes periodically wake up to detect transmissions. To do so, they rely on CCA rather than attempting to pick up a full packet. Unlike X-MAC, senders repeatedly transmit the entire packet for twice the duration of the wake-up period. In case of unicast transmissions, the intended receiver may acknowledge the transmission to notify the sender on correct packet delivery so that the sender can stop transmitting earlier. To implement this functionality, packet transmissions are interleaved with periods of silence in order to allow ACK transmissions. The only LPL parameter tunable by the users is the wake-up period.

CoReDac. CoReDac is a TDMA-based convergecast protocol [5] that builds a collection tree that guarantees collision-free radio traffic. From D-MAC [6] CoReDac borrows the idea of staggered communication. To avoid collisions among packets from their children, CoReDac parents split their reception slots into subslots, and assign one to each child. Packet acknowledgments are pivotal in CoReDac because they piggyback the assignment information, and they are used for synchronizing the TDMA-schedules. A node that misses an acknowledgment must keep its radio on until it hears a new one.

2.2 Clear Channel Assessment

Clear Channel Assessment (CCA) is a mechanism used to determine if a wireless channel is currently free. In wireless MAC protocols, CCA is used to implement Carrier Sense Multiple Access: each node first listens to the medium to detect ongoing transmissions, and transmits the packet(s) only if the channel is free, thus reducing the chance of collisions. CCA is typically implemented by comparing the Received Signal Strength (RSS) obtained from the radio against a threshold. The channel is assumed to be clear if the RSS does not exceed the given threshold. As false negatives result in collisions and false positives cause increased latency, the choice of the threshold is critical [7]. When using CCA to perform CSMA, backoff schemes play an important role. There are two types of backoff: congestion backoff and contention backoff. The former controls the waiting time between consecutive assessments if the channel is not clear. The second controls the waiting time before a retransmission after a collision is detected.

3 Methodology

In our experiments, we use a set of MAC protocols from both the Contiki and TinyOS operating systems. To set a protocol's parameters, we look at the configurations used in popular, low-rate data collection applications [8,9] that employed

similar MAC protocols. These parameters are in general not set to perform optimally under interference.

3.1 Generating Controllable Interference

In our experiments we use a method proposed by Boano et al. [1] to generate customized, controllable, and repeatable interference patterns using common sensornet devices. This method enables the generation of precisely adjustable levels of interference on a specific channel, by exploiting the special test modes of the radio chip.

3.2 Performance Measurements

We use Contiki's software-based power profiler [10] to measure power consumption. For the experiments concerning TinyOS, we have implemented the same mechanism in TinyOS. For computing the power consumption, we assume a current of 20 mA for the radio in receive mode, and a voltage of 3 V, as measured by Dunkels et al. [10]. In all our experiments, the power consumed by the radio in receive mode (*RX power*) is much higher than the one used for transmitting (*TX power*). Because of its strobe mechanism, X-MAC has the highest TX power among the MAC protocols that we examine. At 60% interference, the TX power is around 1 mW, whereas the RX power is almost 20 mW. For LPP instead, the TX power is usually between 0.1 and 0.2 mW only. The power values represent the average power during the full experiment. Since the RX power is at least an order of magnitude larger than the TX power in our experiments, we display only the RX power in our graphs.

3.3 Experimental Setup and Interference Model

In our experiments we put three nodes near each other: a sender, a receiver, and an interferer. The latter interferes using the CC2420's maximum output power level (31), while the sender and the receiver use TX power level 7. The placement of the nodes and their power levels ensure that an active interferer blocks any ongoing communication between the sender and the receiver.

Interference may result from other packet radios (Wi-Fi, Bluetooth, and other sensor networks) operating in the same frequency band, and from other electromagnetic sources such as motors or microwave ovens. Unfortunately, at the time of writing, there are no accepted interference models – an important research issue by itself that is beyond the scope of this paper. Hence, we resort to two simple models here. The *bursty interferer* models continuous blocks of interference with uniformly distributed duration and spacing. This type of interference may be caused, for example, by Wi-Fi or Bluetooth transmissions. The *semi-periodic interferer* also models continuous blocks of interference, but the duration of the periods and their spacing have smaller variance. This type of interference may be caused, for example, by a sensornet performing periodic data collection.

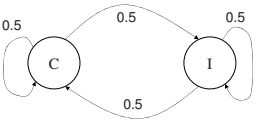


Fig. 2. The interference model used in our experiments

Bursty Interference. In order to describe the transmission and interference patterns, let us define the following random variables:

- S: Bernoulli random variable with parameter 0.5;
- R: Uniformly distributed over [0, 100];
- $Q(x)$: Uniformly distributed over [0, x].

Interference follows continuous off/on periods, and is dictated by a simple two-state discrete Markov process, as depicted in Figure 2. C denotes the clear channel state, and I denotes the interference state. The transitions between the two states is specified by S. At each step of the Markov process, we obtain a time period, $R \times Q(x)$, that determines the duration of the next state. For example, assuming that we move to state I and that we obtain values $R = 40$ and $Q = 20$, the next period will be an interference period of length $40 \times 20 \times 0.3$ ms=240 ms (0.3 ms is a constant factor). $Q(x)$ is used to scale the burstiness of the interference. A higher value represents longer interference slots, such as the ones caused by bursts of Bluetooth or Wi-Fi traffic, whereas a lower value represents shorter transmissions. In the experiments we will select a configuration with long interference slots ($x = 50$) that we call *long bursts*, and a configuration with shorter slots ($x = 8$) that we call *short bursts*.

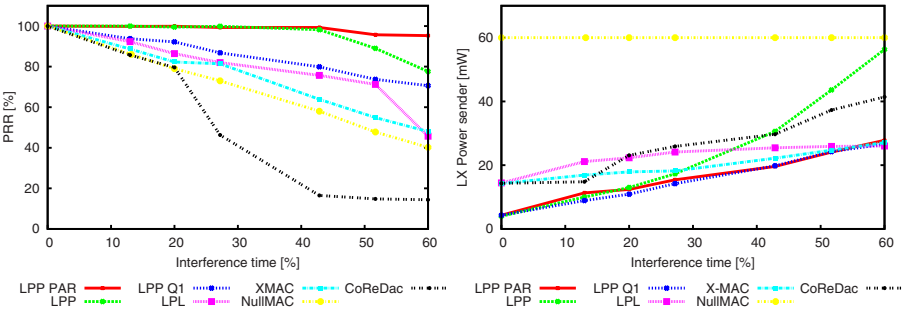
Semi-Periodic Interference. The semi-periodic interferer is a 2-stage process. As described above, we have a clear channel C and an interference I states. The process stays in state I for a time that is uniformly distributed between $\frac{9}{16}$ seconds and $\frac{15}{16}$ seconds. After the transition to state C, it stays in this state for a time that is uniformly distributed between $\frac{3}{4} \times$clear_time and $\frac{5}{4} \times$clear_time, where *clear_time* is a parameter that determines the rate of interference.

4 Experimental Evaluation: The Performance of MAC Protocols under Interference

In this section we report on the performance of several MAC protocols under the different interference patterns described in the previous section.

4.1 Semi-periodic Interference

In our experiments, the sender transmits unicast packets with a payload of 22 bytes to the receiver in a time uniformly distributed between 0.75 s and 1.25 s. We collect the measurements until several thousands packets have been transmitted. We use a semi-periodic interference pattern as described in Section 3.



(a) Packet Reception Rate (b) RX Power Consumption (sender side)

Fig. 3. MAC protocols performance under semi-periodic interference

Figure 3 shows the results of our experiments with different MAC protocols tested against varying interference rates. As expected, the PRR in NULLMAC decreases linearly with the interference rate, following the rule 100% minus the interference rate, which is the probability that a packet is not interfered (Figure 3(a)). The RX power consumption when using NULLMAC is 60 mW independently on the interference pattern, since NULLMAC keeps the radio always on (Figure 3(b)). This confirms the validity of our setup, described in Section 3.

Figure 3(a) shows that all variants of LPP have fairly high packet reception rates compared to the other protocols we consider. Among LPP-based solutions, the best performance is obtained with LPP-PAR, where the receiver transmits a new probe immediately after a packet reception. By doing so, the sender can drain its queue when the interference clears and sustain a high PRR also under high interference by deferring transmissions until interference is over. LPP-PAR outperforms both the standard LPP version, and the so called LPP-Q1, that does not have a queue: a new packet from the upper is discarded in case the previous one has not been transmitted by the MAC layer. At an interference rate of 42%, LPP-Q1 still achieves a PRR of about 80%, showing that even only two probe attempts provide more opportunities to deliver a packet than other solutions.

Figure 3(b) shows that the power consumption of LPP-Q1 is lower than the standard LPP one. The reason comes from the lower PRR shown by LPP-Q1: with fewer packets to be transmitted, the radio is turned off more often. This difference becomes very apparent at an interference rate of 60%, where LPP has its radio turned on almost all the time since there is almost always a packet in the queue waiting to be transmitted. In contrast with the default LPP, LPP-PAR can quickly drain its queue during interference-free periods and hence turn off quickly its radio, saving a substantial amount of power.

X-MAC's packet reception rate is similar but slightly higher than NULLMAC's (Figure 3(a)), since in X-MAC the sender's strobe_time is a little longer than the receiver's off_time. Hence, the receiver has in average more than one chance to hear a strobe. Furthermore, under a semi-periodic interference pattern,

it is unlikely that interference comes into effect during the exchange of strobe, acknowledgment, and data packet, which take very little time. Therefore, if the strobe succeeds, the entire operation most likely successfully completes. The same reasoning also applies for CoReDac when the interference rate is 20% or lower. At higher interference, however, CoReDac looses synchronization and its performance drastically degrades.

With regard to LPL, we observe two modes of operations along the PRR axis in Figure 3(a). When the interference rate is lower than 60%, the CCA mechanism is reasonably effective at detecting the presence of interference, and packet losses occur mostly because of data corruption during the transmission. Indeed, we verify that an increasing number of packets are received but do not pass the integrity checks. The increasing power consumption shown for LPL in Figure 3(b) is simply an effect of the decreasing PRR: the fewer packets are received, the less likely is the sender to receive the acknowledgment and stop the transmissions earlier. On the other hand, at 60% interference it is often the case that the CCA mechanism never finds the channel free. After a maximum number of reattempts, the packet is dropped on the sender side, causing a drastic decrease in PRR. However, without even transmitting the packet, not much energy is spent on the sender side. This is confirmed in Figure 3(b), where the power consumption at 60% interference is still comparable to other settings.

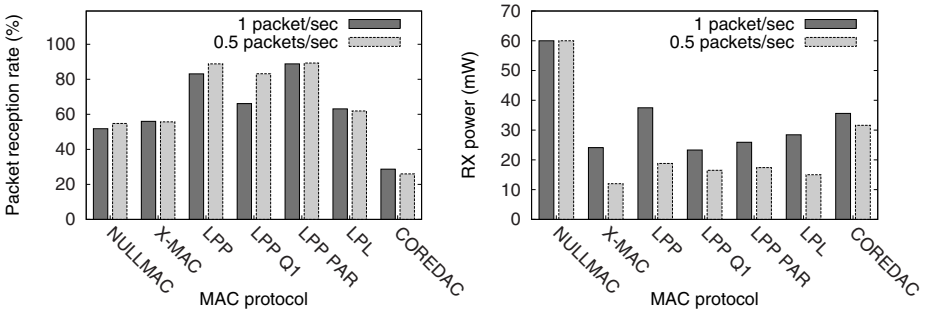
Our results suggest that more handshakes opportunities improve the PRR in interfered networks. When comparing different LPP versions with each other, we can see that we can achieve a low power consumption and a high PRR using LPP-PAR, thanks to its queue drain when a period of interference has ended.

Impact of Queue Size on Performance. Our experiments clearly show that the queue size may drastically change the performance of a MAC protocol under interference. We investigated the impact of the queue size both on power consumption and packet reception rate by running LPP with different queue sizes under 60% semi-periodic interference. Our results show that a queue size of four packets guarantees good performance.

4.2 Bursty Interference

We carry out the same set of experiments in presence of bursty interference ($x = 50$, see Section 3), and different transmission rates, in order to investigate how performance changes depending on the network load. Figure 4 illustrates the results.

For most MAC protocols the PRR does not change depending on the transmission rate (Figure 4(a)). In most cases, indeed, the interference rate is what ultimately determines the observed PRR. An exception is LPP-Q1, where the PRR increases by almost 10% when the application transmits packets less frequently. The reason is that with higher transmission rates, a packet cannot be sent before the application hands the next packet to the MAC layer, and thus the latter packet is discarded. This can either happen with long periods of interference, or when periods of interference overlap with the instants in which the receiver sends probes.



(a) Packet Reception Rate (b) RX Power Consumption (sender side)

Fig. 4. MAC protocols performance under bursty interference

5 The Impact of Clear Channel Assessment and Congestion Backoff under Interference

While many contention-based MAC protocols implement CSMA, one could also start transmitting a packet without carrying out CCA. The latter approach saves the CCA overhead of listening to the channel and switching the radio between send and receive modes, which may take hundreds of microseconds [11]. Few retransmissions consume a negligible amount of power compared to a continuous use of CCA. An increased probability of collisions may be negligible in low data rate applications, but not in settings with high interference.

A second aspect that affects the performance of CSMA-based MAC protocols such as B-MAC [12], WiseMAC [13], and BoX-MAC [14] is the backoff algorithm that adapts the scheduling of CCA executions to wireless channel conditions. B-MAC, for example, uses by default a small random congestion and contention backoff time, but does also support user-defined backoff schemes. BoX-MAC uses a randomized long congestion backoff period in the order of a few hundred milliseconds.

In this section we identify (1) the scenarios where adopting CCA improves or decreases the performance of MAC protocols under interference, and (2) if the choice of the congestion backoff scheme plays a pivotal role under interference. We investigate these issues in terms of energy efficiency and latency.

5.1 Experimental Setup

In our first experiment, we compare a scenario in which CCA is not used (and packets are sent without a carrier sense) with one in which a node sleeps after detecting a busy channel for a congestion backoff time B_C. We explore different types of backoff algorithms, in particular null (no waiting time), constant (waiting time uniformly drawn from a fixed backoff window), linear (backoff window increases by a constant amount after failed CCA), quadratic (backoff window squared after failed CCA), and cubic (backoff window cubed after failed CCA).

We select an initial backoff time randomly short and we eventually increase it according to the backoff algorithm. We further study a variant where the backoff is truncated after $R = 8$ CCA attempts. We use the CC2420's default CCA threshold.

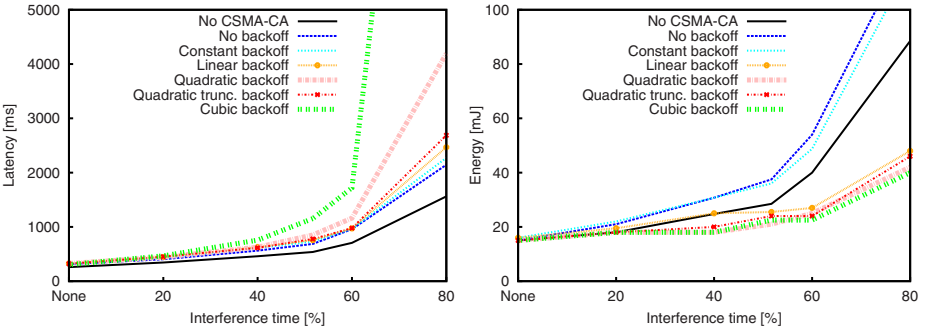
Our experimental setup is described in Section 3. The transmitter sends N packets towards the receiver at different transmission rates. Each packet has to be acknowledged within $\frac{1}{64}$ seconds.

We further investigate two different strategies for scheduling retransmissions. With the first approach, queued packets are retransmitted immediately after timeout. With the second approach, the sender turns off its radio after a timeout occurs, and the queued packets are retransmitted according to the original packet transmission rate (e.g. after 0.5 seconds if we transmit 2 packets per second). We measure the latency required to transmit the sequence of N packets and the total amount of energy consumed by the radio of the sender. The latter is appropriate because interference mainly affects the sender, assuming that a receiver can distinguish valid data from interference and go back to sleep in case of the latter. The sender node runs NULLMAC with or without CSMA, and its radio is turned off after the reception of an ACK (or after the timeout fires), and turned on again for the next transmission. Since we are only interested in the energy consumption of the sender, the receiver keeps the radio on all the time. To isolate the effect of CCA from that of other MAC mechanisms, we avoid mechanisms such as LPL and the associated use of long preambles.

5.2 Experimental Results

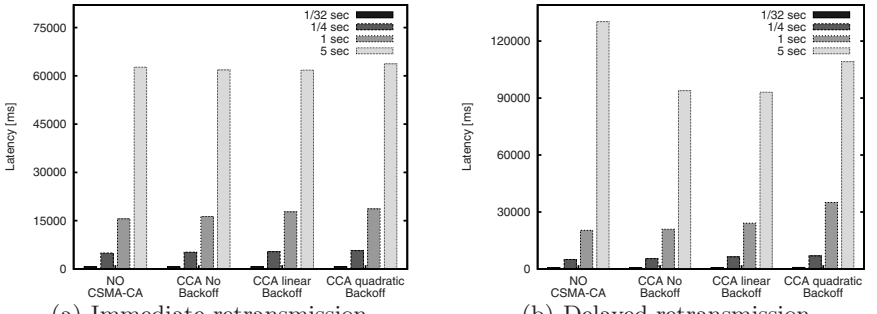
In the first set of experiments, we evaluate the communication performance when transmitting $N = 50$ packets at the highest available rate, and compare transmissions with and without CSMA. We average the results after sending several thousand packets. Figure 5 shows the results. As expected, the more aggressive the backoff strategy is, the lower is the energy required to complete the transmission. The latency increases proportionally with the backoff delays, however, indicating a tradeoff between energy consumption and latency. The energy consumption is, however, significantly reduced when not using CSMA, but using aggressive backoffs such as quadratic and cubic algorithms on a channel that is interfered more than 20% of the time. We can also see that truncating the backoff window yields a good balance between energy and latency.

In the scenario presented above, the packets are retransmitted as soon as the timeout event occurs. If queued packets are retransmitted back-to-back under interference, there is a significant waste of energy due to the medium still being busy, while a retransmission based on the original transmission rate increases the overall latency. To quantify these issues, we carry out another experiment with different periodic transmission rates. We transmit bursts of $N = 10$ packets with and without CSMA, using null, linear, and quadratic congestion backoff schemes. Then we apply a bursty interference pattern with long bursts ($x = 50$) and measure the latency and energy consumption at the sender side, averaging the results of several hundred bursts.



(a) Latency (b) Energy consumption

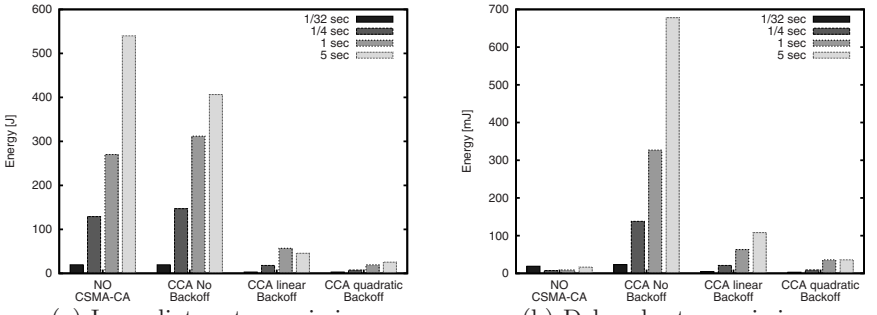
Fig. 5. Energy consumption and latency measured at the sender side, when sending bursts of $N = 50$ packets at the highest rate available.



(a) Immediate retransmission (b) Delayed retransmission

Fig. 6. Latency measured at the sender side when sending bursts of $N = 10$ packets at different transmission rates, with different retransmissions schemes.

Figures 6 and 7 show the results. As expected, if queued packets are retransmitted back-to-back, the approach without CSMA performs poorly. A configuration with quadratic congestion backoff requires only 5% of the energy used without CSMA with an acceptable latency because of the fewer attempts. If, instead, queued packets are retransmitted according to the original transmission rate, the protocol that does not adopt CSMA performs better in terms of energy efficiency. This is because it attempts to transmit only at the instants defined by the transmission rate, while the approach with CSMA and backoff tries to find the first instant at which the medium is free, often without success. This makes the approach without CSMA more energy-efficient, but comes with an increased latency when sending at low transmission rates, such as one packet every 5 seconds w.r.t. CSMA transmissions. As in the previous experiment, a more aggressive congestion backoff scheme such as the quadratic algorithm shows a good balance between latency and energy consumption.



(a) Immediate retransmission (b) Delayed retransmission

Fig. 7. Energy consumption measured at the sender side, when sending bursts of $N = 10$ packets at different transmission rates, with different retransmissions schemes.

In addition to the above experiments with long bursts, we also carried out experiments with shorter bursts ($x = 8$, see Section 3). Due to space constraints we do not show the results here. These experiments indicate a better performance of protocols using CSMA, because shorter slots will imply a lower energy consumption since the channel will be sampled a smaller amount of times.

In conclusion, our experiments demonstrate that the choice of congestion backoff scheme plays a pivotal role for MAC protocols that use CCA. These results act as a guideline for protocol designers. A CSMA approach with a quadratic backoff –truncated or not– performs well in most scenarios.

6 Improvements

The results presented in Section 4 show two methods that can make MAC protocols more robust against interference: (1) holding a packet longer so that multiple handshake attempts are possible, and (2) implementing packet trains as a means to quickly send multiple packets that have accumulated during interference. Section 5 further shows that the power consumption can be reduced by applying suitable congestion backoff schemes when using CCA. We extend the X-MAC implementation in Contiki 2.3 with these mechanisms, and evaluate it under random interference patterns.

6.1 Design and Implementation of a Robust X-MAC

We design a new version of X-MAC, called X-MAC/Q, that is able to maintain high packet reception rates and low power consumption despite being challenged by interference. The new version contains a packet queue implemented by using a statically allocated array of packets and their corresponding attributes. By default, the queue stores up to four packets, the optimal value for LPP as discussed in Section 4.1. Since only unicast packets are acknowledged in the X-MAC protocol implementation, we only queue unicast packets.

Packet Queue with Fast Drain. Unlike the original implementation of X-MAC in Contiki, our augmented implementation revolves around the packet queue. This distinction starts from the existing packet transmission method, *qsend_packet()*, where all unicast packets are put into a queue. The packets will not be sent directly, but instead linger shortly for a configurable time ($\frac{1}{32}$ s in our experiments.) The linger time makes it possible to accumulate packets into the queue, which allows the layer on top of X-MAC to create a burst of packets. When the accumulation timer has expired, X-MAC/Q gets the oldest packet from the queue, and immediately starts sending strobes to the addressed receiver of the packet. To enable fast queue draining, each strobe contains the amount of packets for the destination that the sender has in its queue. If the sender receives a strobe acknowledgment within a configured waiting time, it sends one packet at a time, including the strobe procedure, separated by a very short time ($\frac{1}{128}$ s) instead of the usual duty-cycle interval. If the sender does not receive the strobe acknowledgment, a new attempt comes after $\frac{1}{32}$ s. Packets are removed from the queue when they have either been successfully sent, or timed out after 10 s. The X-MAC reception method requires only two changes. First, each received strobe will contain the amount of packets x that the receiver should receive in a train. Second, the receiver stays awake until it has received x packets since the strobe.

Clear Channel Assessment with Congestion Backoff. Based on the results in Section 5, we extend X-MAC/Q to include clear channel assessments with a linear and a quadratic congestion backoff timers. The version with the linear backoff is called X-MAC/QL, whereas the version with quadratic backoff is called X-MAC/QQ. Before sending out the first strobe the new versions turn on the CCA to check if the channel is clear. If the CCA check fails, we wait for ($\frac{1}{128}$ × number_of_attempts) or ($\frac{1}{128}$×number_of_attempts^2) milliseconds before another attempt for X-MAC/QL and X-MAC/QQ respectively.

6.2 Experimental Evaluation

We repeat the experiments with the bursty interferer described in Section 4.2 using our improved versions of X-MAC. For comparison, we also show the LPP-PAR and another X-MAC improvement that we call X-MAC/LT. X-MAC/LT is similar to X-MAC except for one parameter, *strobe_time*, which we increase from offtime + 20 × ontime to 4 × offtime + 20 × ontime. Because X-MAC/LT holds packets longer, we expect a higher PRR compared to X-MAC.

Figure 8 shows that both X-MAC/Q and X-MAC/LT significantly increase the PRR compared to the default X-MAC. When the applications send one packet every two seconds, the PRR is similar to the one of LPP-PAR. Also, both new X-MAC versions show a similar rate, but the left graph in Figure 8 shows that the power consumption is much higher for X-MAC/LT than for X-MAC/Q. X-MAC/QQ and X-MAC/QL achieve a good PRR with very low power consumption. Since both protocols wait for an increasing amount of time when the medium is kept busy, they send less strobes and avoid to wait for

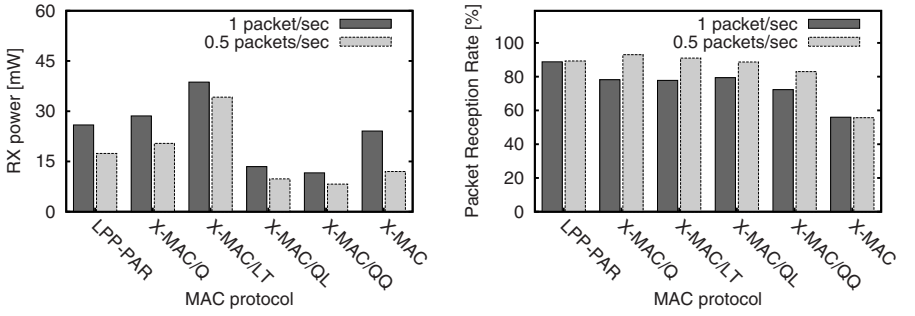


Fig. 8. Our experiments show that the proposed mechanisms increase the robustness of X-MAC to interference

strobe acknowledgments that will not arrive, thus saving a significant amount of power. Compared to X-MAC/QQ, X-MAC/QL consumes slightly more energy but achieves a higher PRR. This follows the results presented in Section 5.2: the linear backoff causes more frequent samples of the channel than the quadratic one does, leading to higher power consumption. On the other hand, the quadratic algorithm may grow its sampling interval exponentially up to a point where expired packets will be removed from the queue.

In all our experiments, we set the protocol parameters based on the configurations of similar MAC protocols in popular applications [8,9], since our goal is not to optimize parameters but to identify mechanisms that enable good performance during interference. One way of increasing the handshake frequency would be to change the parameters. In X-MAC, this is the *offtime* parameter. We have rerun the same experiment as in Figure 8, but halved the *offtime* to 1/4 s for X-MAC and X-MAC/Q. Our results show similar improvements in PRR and power consumption for both protocols. For the CCA versions with a linear backoff, the improvements of the PRR were smaller but the power consumption was decreased by around 40%.

In summary, our results show significant improvements of the packet reception rate for X-MAC/Q with a moderate increase in power consumption. X-MAC/QQ and X-MAC/QL's power consumption is even lower than X-MAC's despite that they achieve a much higher PRR.

7 Related Work

Radio interference has been a topic of significant interest in the sensor network community. Most of the earlier work focused on deriving fair transmission schedules by synchronizing the transmission of neighboring nodes in the presence of interference [15,16,17,18]. Our work also addresses MAC performance, but our goal is to identify experimentally some mechanisms that improve the robustness of MAC protocols against interference.

Zhou et al. present some important differences between the interference behavior of real and ideal scenarios [19,20]. Others study interference effects on real deployments: Rangwala et al. propose an interference-aware fair-rate control evaluated on real hardware [21]. Others have proposed frequency hopping solutions for 802.15.4 networks in order to overcome Wi-Fi interference [22,23].

Motivated by the empirical works mentioned above, we (1) analyze experimentally the impact of interference on various MAC protocols, and (2) propose mechanisms to increase packet delivery rate and reduce energy consumption.

An important group of work pertaining to this study is the set of notable MAC protocols evaluated on empirical testbeds, in particular X-MAC[2], LPP[3], LPL[12]. Most of these evaluations focused on energy efficiency and delay under different traffic patterns while we evaluate the protocols behaviour under various degrees of interference. Bertocco et al. investigate efficient CCA thresholds in presence of in-channel wide-band additive white Gaussian noise [7]. In this work, we study the role of CCA and congestion backoff schemes with respect to energy consumption and latency under generic patterns of interference. So far, thorough studies on backoff schemes have been performed only with respect to contention resolution [24], [25], and [26], where Jamieson et al. propose a MAC protocol that uses a fixed-size contention window and a non-uniform probability distribution of transmitting in each slot within the window.

Moss and Levis envisioned how a long congestion backoff could at the same time optimize energy and delivery rates in congested networks [14]. However, they do not determine optimal backoff periods and do not quantify the effects of different schemes. We demonstrate experimentally the impact of the congestion backoff time on energy efficiency and latency in networks with high interference.

8 Conclusions

In this paper, we experimentally study the impact of interference on several MAC protocols. Using the results from our experiments, we identify mechanisms that make MAC protocols more robust against interference. We augment an existing X-MAC implementation with these mechanisms, and demonstrate improved packet reception rates and reduced power consumption in cases where the radio communication is challenged by interference.

Acknowledgments

This work has been partially supported by the European Commission under the contract No. FP7-2007-2-224053 (CONET). This research has been also partially financed by VINNOVA, the Swedish Agency for Innovation Systems, by SSF, and by the IRCSET Postdoctoral fellow Grant PD200857, SFI Grant No. SFI08CEI1380. This work has been partially supported by the Cluster of Excellence 306/1 "Inflammation at Interfaces" (Excellence Initiative, Germany).

References

1. Boano, C.A., He, Z., Li, Y., Voigt, T., Zuniga, M., Willig, A.: Controllable Radio Interference for Experimental and Testing Purposes in Wireless Sensor Networks. In: Proc. of the 4th Workshop on Practical Issues in Building Sensor Network Applications (SenseApp), Zurich, Switzerland, October 2009. IEEE Computer Society, Los Alamitos (2008)
2. Buettner, M., Yee, V., Anderson, E., Han, R.: X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In: ACM SenSys (2006)
3. Musaloiu-E., R., Liang, C.-J.M., Terzis, A.: Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. In: IPSN 2008 (2008)
4. TinyOS Community Forum. TinyOS TEP 126 - CC2420 radio stack, http://www.tinyos.net/tinyos-2.x/doc/html/tep126.html
5. Voigt, T., Österlind, F.: CoReDac: Collision-free command-response data collection. In: 13th IEEE Conference on Emerging Technologies and Factory Automation, Hamburg, Germany (September 2008)
6. Lu, G., Krishnamachari, B., Raghavendra, C.: An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks. In: International Parallel and Distributed Processing Symposium, IPDPS (2004)
7. Bertocco, M., Gamba, G., Sona, A.: Experimental Optimization of CCA Thresholds in Wireless Sensor Networks in the Presence of Interference. In: Proc. of IEEE Workshop on ElectroMagnetic Compatibility (IEEE EMC) (June 2007)
8. Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., Anderson, J.: Wireless sensor networks for habitat monitoring. In: First ACM Workshop on Wireless Sensor Networks and Applications (WSNA 2002) (September 2002)
9. Tolle, G., et al.: A macroscope in the redwoods. In: SenSys, pp. 51–63 (2005)
10. Dunkels, A., Österlind, F., Tsiftes, N., He, Z.: Software-based on-line energy estimation for sensor nodes. In: Proceedings of the 4th Workshop on Embedded Networked Sensors (EmNetS), Cork, Ireland (June 2007)
11. Chintalapudi, K.K., Venkatraman, L.: On the design of mac protocols for low-latency hard real-time discrete control applications over 802.15.4 hardware. In: The 7th Conf. on Information Processing in Sensor Networks (IPSN) (April 2008)
12. Polastre, J., Hill, J., Culler, D.: Versatile low power media access for wireless sensor networks. In: ACM SenSys (2004)
13. El-Hoiydi, A., Decotignie, J.D.: Wisemac: An ultra low power mac protocol for the downlink of infrastructure wireless sensor networks. In: ISCC (June 2004)
14. Moss, D., Levis, P.: Box-macs: Exploiting physical and link layer boundaries in low-power networking. Technical Report SING-08-00, Stanford University (2002)
15. Tassiulas, L., Sarkar, S.: Maxmin fair scheduling in wireless ad hoc networks. IEEE Journal on Selected Areas in Communications 23(1), 163–173 (2005)
16. Chen, L., Low, S.H., Doyle, J.C.: Joint congestion control and media access control design for ad hoc wireless networks. In: INFOCOM (2005)
17. Yi, Y., Shakkottai, S.: Hop-by-hop congestion control over a wireless multi-hop network. IEEE/ACM Transactions On Networking 15(1), 133–144 (2007)
18. Yi, Y., de Veciana, G., Shakkottai, S.: On optimal MAC scheduling with physical interference. In: INFOCOM (2007)
19. Zhou, G., He, T., Stankovic, J.A., Abdelzaher, T.: RID: Radio interference detection in wireless sensor networks. In: INFOCOM (2005)
20. Zhou, G., et al.: Models and solutions for radio irregularity in wireless sensor networks. ACM Trans. Sen. Netw. 2(2), 221–262 (2006)

21. Rangwala, S., Gummadi, R., Govindan, R., Psounis, K.: Interference-aware fair rate control in wireless sensor networks. In: ACM SIGCOMM (2006)
22. Musaloiu-E, R., Terzis, A.: Minimising the effect of wifi interference in 802.15.4 wireless sensor networks. Journal of Sensor Networks 3, 43–54 (2007)
23. Hauer, J., Handziski, V., Wolisz, A.: Experimental study of the impact of wlan interference on ieee 802.15.4 body area networks. In: Roedig, U., Sreenan, C.J. (eds.) EWSN 2009. LNCS, vol. 5432, pp. 17–32. Springer, Heidelberg (2009)
24. Yuan, Z., et al.: A backoff copying scheme for contention resolution in wireless sensor networks. In: Wintech (September 2009)
25. Athanasopoulos, A., et al.: 802.15.4: The effect of different back-off schemes on power and qos characteristics. In: IEEE ICWMC (2007)
26. Jamieson, K., Balakrishnan, H., Tay, Y.C.: Sift: a mac protocol for event-driven wireless sensor networks. In: Römer, K., Karl, H., Mattern, F. (eds.) EWSN 2006. LNCS, vol. 3868, pp. 260–275. Springer, Heidelberg (2006)

MaxMAC: A Maximally Traffic-Adaptive MAC Protocol for Wireless Sensor Networks

Philipp Hurni and Torsten Braun

Institute of Computer Science and Applied Mathematics
University of Bern
{hurni,braun}@iam.unibe.ch

Abstract. Energy efficiency is a major concern in the design of Wireless Sensor Networks (WSNs) and their communication protocols. As the radio transceiver typically accounts for a major portion of a WSN node's power consumption, researchers have proposed Energy-Efficient Medium Access (E^2-MAC) protocols that switch the radio transceiver off for a major part of the time. Such protocols typically trade off energy-efficiency versus classical quality of service parameters (throughput, latency, reliability). Today's E^2-MAC protocols are able to deliver little amounts of data with a low energy footprint, but introduce severe restrictions with respect to throughput and latency. Regrettably, they yet fail to adapt to varying traffic load at run-time.

This paper presents MaxMAC, an E^2-MAC protocol that targets at achieving maximal adaptivity with respect to throughput and latency. By adaptively tuning essential parameters at run-time, the protocol reaches the throughput and latency of energy-unconstrained CSMA in high-traffic phases, while still exhibiting a high energy-efficiency in periods of sparse traffic. The paper compares the protocol against a selection of today's E^2-MAC protocols and evaluates its advantages and drawbacks.

Keywords: Wireless Sensor Networks, Energy Efficient Medium Access Control, Traffic Adaptivity.

1 Introduction

Today's E^2-MAC protocols generally reduce the power consumption at the cost of deteriorating quality of service, in particular by an increase of packet latency and a decrease of throughput and reliability. In the tradeoff between energy and quality of service, researchers have concentrated almost exclusively on the energy aspect, introducing tight restrictions with respect to throughput and latency. Such restrictions may be tolerable in networks with low quality of service requirements. However, many event-based scenarios require reasonable quality of service during periods of increased activity, and a high energy-efficiency during long periods of inactivity. Such scenarios can be found e.g. in monitoring systems for healthcare [1], in Disaster-Aid-Systems [2], but also in the broad area of (event-based) environmental monitoring systems. Varying, temporarily high

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 289–305, 2010.

traffic can further be expected to appear in the emerging field of multimedia sensor networks (WMSNs) [3]. Once an event has been triggered, e.g. a patient's pulse monitor registering anomalies in a hospital or geriatric clinic, the MAC protocol's primary objective should shift towards delivering good quality of service (high throughput, low delay) rather than saving energy. In such scenarios, today's E^2-MAC protocols do not provide reasonable flexibility, as most of them were designed under the assumption of very sparse low-rate traffic.

This paper introduces MaxMAC, an energy-efficient MAC protocol for sensor networks designed for WSN scenarios with varying traffic conditions. While MaxMAC operates similarly as existing E^2-MAC protocols in low traffic situations, it is able to maximally adapt to changes in the network traffic load at run-time. Taking advantage of design principles for E^2-MAC protocols developed over the last couple of years, the protocol introduces novel run-time adaptation techniques to effectively allocate the costly radio transceiver truly in an *on demand* manner. The protocol reaches the throughput and latency of energy-unconstrained CSMA in situations of high-traffic, yet exhibiting a high energy-efficiency in periods of sparse traffic.

The paper is organized as follows: Section 2 discusses related work on the topic of traffic-adaptive E^2-MAC protocols. Section 3 then describes the design of the MaxMAC protocol mechanisms. Section 4 presents simulation setup and environment, followed by simulation results in Section 5. Section 6 concludes the paper.

2 Related Work

A couple of concepts has yet been applied to reach traffic-adaptive protocol behavior in today's literature on E^2-MAC protocols. However, most approaches are minor variations of existing protocols and still heavily restrain throughput and latency of the MAC layer, a crucial disadvantage which often prevents them to be applied in real WSN deployments.

T-MAC [4] increases the traffic-adaptivity of S-MAC [5] by prolonging the duty cycles of the nodes when so-called activation events occur. An activation event may be the sensing of any communication in the neighborhood, the end of the own data transmission or acknowledgement, the overhearing of RTS or CTS control messages that may announce further packet exchanges. However, simulations show that the adaptivity of the protocol is still very limited and that the performance gain of the traffic adaptivity enhancement further only pays off for non-uniform bursty traffic.

X-MAC [6] is an E^2-MAC protocol based on asynchronous listen-intervals. For each packet, X-MAC transmits a strobe of preambles, in between which the receiver can signal reception-readiness with a so-called *EarlyACK*. [6] derives a formula for optimal wake/sleep intervals given traffic at a certain rate and outline a mechanism to let X-MAC adapt the duty cycle and the sleep/wake interval to best accommodate the traffic load in the network. With the basic mechanism of X-MAC still requiring a certain minimal interval between two active intervals

and a generally high per-packet overhead, the maximum achievable throughput of the protocol remains very limited.

AMAC [7] is an E^2-MAC protocol targeting at traffic-awareness. It relies on the S-MAC active period structure consisting in SYNC, RTS and CTS windows. With low traffic, AMAC neglects the costly RTS/CTS exchange and operates with a large sleep interval between two active periods. With increasing traffic, it multiplies the amount of active periods by a factor of 2^n, thus increasing the net duty cycle by the same factor. Applying this adaptation strategy, the protocol can prevent packet drops to some extent while still saving energy.

Z-MAC [8] is a TDMA-based protocol that achieves high channel utilization under high contention. The protocol initially gathers topology information and rigidly synchronizes clocks to maintain a collision-free schedule. Under low traffic, its performance with respect to energy-efficiency however remains low.

BurstMAC [9] is a recent E^2-MAC protocol targeting at achieving a low idle-overhead *and* a high throughput in case of correlated traffic bursts, as they occur in event-based scenarios. BurstMAC employs multiple channels and keeps a rigid network-wide synchronization and TDMA-scheme, The protocol achieves high throughput in case of correlated event traffic by efficient on-demand allocation of channels, hence letting node pairs communicate concurrently.

3 MaxMAC Design

3.1 Basic Media Access Mechanism

Many energy-efficient protocol mechanisms for wireless sensor MAC protocols have been developed during the past couple of years. MaxMAC takes advantage of the substantial work carried out on E^2-MAC protocols, especially the asynchronous protocols B-MAC [10], WiseMAC [11] and X-MAC [6]. This section briefly discusses the basic media access mechanisms used in MaxMAC, while Section 3.2 discusses its run-time traffic adaptation mechanisms.

Preamble Sampling: With Preamble Sampling (also referred-to as *Low-Power-Listening*) introduced in B-MAC and WiseMAC, nodes keep their radios off for most of the time and only wake up for very brief periodic duty cycles to poll the channel for a preamble signal. The sender node prepends a preamble for each frame that signals the upcoming frame transmission to the receiving node in its short wake-up. In B-MAC, the preamble spans the entire wake-up interval, whereas WiseMAC learns the wake-up schedules of its neighbors to minimize the length of the preambles in future transmissions. A small preamble then only compensates for the maximum clock drift that the two involved node's clocks may have developed during the time since the last schedule exchange. Given that digital crystal oscillators typically exhibit low drifts ($\leq$ 100 ppm), this preamble minimization scheme incurs a low per-packet overhead while still achieving a high packet delivery probability. MaxMAC takes advantage of the WiseMAC preamble-sampling scheme - each node periodically wakes up to sense the channel for a preamble tone within the *Base Interval* T (cf. Figure 1).

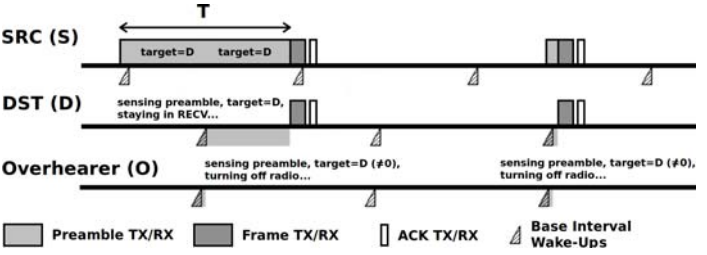


Fig. 1. Preamble sampling with embedded target address in MaxMAC

Overhearing Avoidance: The preamble sampling technique of WiseMAC is already quite efficient in avoiding costly overhearing. With sparse traffic, chances are high that the wake-ups of non-targeted receivers do not coincide with those of the target receivers. With higher traffic, however, and transmissions of queued packet trains, overhearing of preambles and frames becomes an increasing source of energy waste. MaxMAC minimizes overhearing by enriching preambles with target id information, as illustrated in Figure 1. Target nodes turn their radio transceiver on, sense the carrier for *their particular preamble* to receive preamble and frame. Non-target nodes turn their radios on, extract the target information in the ongoing preamble transmission, notice that they are not targeted and immediately turn it off. This concept has been applied in X-MAC [6], where nodes send preamble strobes in-between which receiver nodes can signal reception readiness with a so-called *Early-ACK*. MaxMAC however applies this concept to reduce overhearing in a preamble-sampling MAC protocol, combined with the preamble minimization technique of WiseMAC [11].

3.2 Run-Time Traffic Adaptation Mechanisms

In contrast to most of today's E^2-MAC protocols, which operate with rather static parameter settings, MaxMAC introduces traffic-adaptation features to instantly react to changing load conditions by altering it's behavior at run-time. MaxMAC attempts to allocate the energy resources of the sensor node in an *on-demand* manner. Similarly as in dynamic frequency/voltage scaling, where the CPU reacts to higher computation load with an increase of the frequency/voltage, a traffic-adaptive E^2-MAC protocol should react to changing load conditions by correspondingly tuning the radio transceiver - turning/keeping the transceiver on more frequently when more traffic has to be handled, keeping it permanently on during load peaks, and turning it off again when the load level permits it.

Allocation/Deallocation of Extra Wake-Ups: With E^2-MAC protocols alternating between sleep and wake intervals, throughput is often restrained to a couple of frame transmissions in each interval. Latency typically increases sharply, as forwarding nodes need to buffer incoming frames and wait for the next wake-up of their gateway node, which often sums up to some seconds in

multi-hop scenarios. The first traffic adaptation feature and essential novelty of MaxMAC tackles this very decisive E^2-MAC protocol restriction. In MaxMAC, nodes change their state (and hence their behavior) and allocate so-called *Extra Wake-Ups* when the rate of incoming packets reaches predefined threshold values, and de-allocate them when the rate drops below the threshold again.

Figure 2 illustrates the state-based adaptivity mechanism with a source node (SRC) sending packets to a receiver node (DST) with increasing rate. Nodes operate in the *Base Interval* state per default, polling the channel periodically within the Base Interval T. Nodes alter their state (and behavior) by switching to states S_1, S_2 when the corresponding thresholds T_1, T_2 are reached. Thresholds T_1 and T_2 are set to 2 and 6 packets/s in the illustration in Figure 2. Each node keeps estimating the rate of incoming packets, using a sliding window of 1s (cf. rate-estimation graph of DST in Figure 2). With the rate of incoming packets reaching the threshold T_1, the DST schedules one additional *Extra Wake-Up* in-between each Base Interval, effectively doubling the amount of duty cycles over time. The receiver node DST communicates its increased wake-up frequency in the ACK. SRC receives this announcement and marks the increased wake-up frequency of node DST in its schedule offset table. With the notification sent by DST in the ACK, DST *promises* to remain in the new state and keep its increased wake-up frequency for a predefined timespan S1_LEASE. For each state in MaxMAC, the LEASE timespans (S1_LEASE, S2_LEASE, CSMA_LEASE) define how long a node *promises* to remain in the new state when announcing the state change in the ACK. LEASE timespans can further be *prolonged* in any new ACK transmission. By remaining in a higher state for at least the LEASE duration, fast oscillation between the different states can be mitigated. With the rate of incoming packets reaching the threshold T_2, DST changes to state S_2, doubles the amount of wake-ups again and announces its state change in the ACK (cf. Figure 2). As soon as these timespans expire, nodes having received prior state change announcements will assume that the corresponding node has fallen back to its default behavior (polling the channel with the Base Interval T),

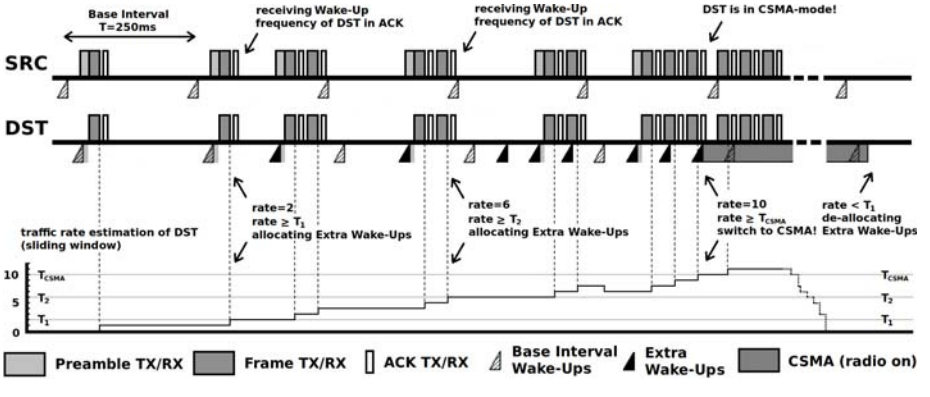


Fig. 2. Adding Extra Wake-Ups with increasing rate of traffic

which prevents them from transmitting at instants when the target is not awake. All LEASE timespans are set to 1s in the subsequent experiments.

Increasing the amount of wake-ups is an effective, yet considerably cheap means of increasing network throughput and decreasing end-to-end latency. If SRC needs to forward other packets, the time to wait for the next wake-up of DST is halved with DST being in state S_1 or even quartered with DST being in state S_2. However, if the additional wake-ups scheduled by DST are not used for transmissions, the waste of energy remains limited, as some few additional channel polls are energetically inexpensive.

Exploiting the Channel Capacity by switching to CSMA: Most existing E^2-MAC protocols have been designed under the assumption of sparse low-rate traffic. Hence, these protocols severely restrain throughput, compared to energy-unconstrained wireless channel protocols. In multi-hop scenarios, S-MAC, T-MAC and WiseMAC have been shown to reach only a fraction of that of CSMA [12] [13]. MaxMAC has been specifically designed to achieve a throughput similar as CSMA in situations of increased network activity, after a certain delay for triggering the adaptation mechanisms. While the allocation of *Extra Wake-Ups* helps to achieve a somewhat increased throughput, CSMA-like throughput and latency can not yet be reached with it. MaxMAC thus carries the threshold-based concept one step further. When the rate of incoming packets reaches a further threshold T_{CSMA} (with $T_{CSMA} > T_2 > T_1$), MaxMAC switches to energy-unconstrained CSMA and announces this state change to the sender node (and potentially overhearing child nodes) in the ACK. Figure 2 illustrates node DST measuring the rate of incoming packets to reach $T_{CSMA} = 10$ packets/s in the right part of the figure. Node DST hence switches to the CSMA state, announcing the state change to SRC in the ACK, hence *promising* to remain in the CSMA state for at least the predefined timespan CSMA_LEASE. Within this timespan, SRC can transmit packets without having to wait for a wake-up of DST, as it *knows* that DST keeps its transceiver on for at least the timespan CSMA_LEASE. With CSMA_LEASE expiring, all nodes having received the prior state change announcement of DST assume that DST has fallen back to the Base Interval state, which prevents them from transmitting at times when DST is asleep.

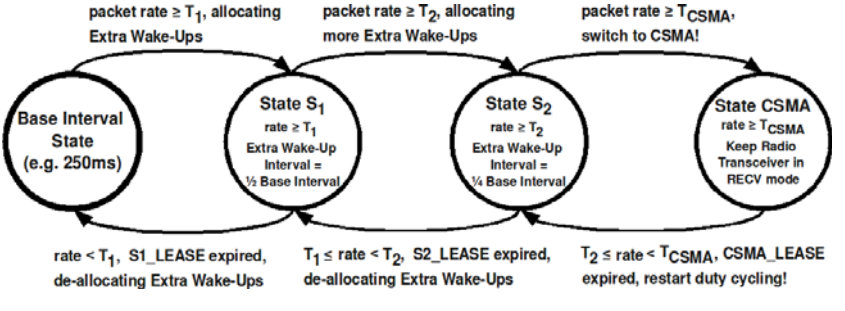


Fig. 3. State-based traffic adaptivity mechanism of MaxMAC

Figure 3 illustrates the state-based adaptivity concept of MaxMAC with the state transitions as a finite state machine. Nodes switch from the Base Interval state to a higher state $S_1, S_2, CSMA$ when the rate reaches the associated thresholds T_1, T_2, T_{CSMA}. When switching from the Base Interval state to S_1 or S_2, nodes schedule *Extra Wake-Ups* and double or quadruple their wake-up frequency, which increases network throughput and reduces end-to-end latency. When the rate reaches the threshold T_{CSMA}, nodes switch to energy-unconstrained CSMA and keep their radio transceivers turned on. With the load falling below T_{CSMA} and CSMA_LEASE expiring, nodes switch again to states S_1 or S_2 and restart alternating between brief channel polls and long sleep intervals. Nodes completely de-allocate all *Extra Wake-Ups* and fall back to the Base Interval state when the packet rate drops below T_1 and all LEASE timespans have expired. The MaxMAC traffic adaptation mechanism scales well for multi-hop topologies, as each node measures and reacts upon a given rate increase in a decentralized manner. MaxMAC further communicates state changes efficiently, without introducing any new control messages. All the necessary control information is communicated in the Data frame header and the ACK frames.

This section illustrates the MaxMAC adaptivity concept with three states S_1, S_2, $CSMA$ - the number of states and thresholds can however be chosen arbitrarily. The threshold values T_1, T_2, T_{CSMA} we choose in Section 5 were calibrated for the particular given scenarios. The thresholds allow the network operator for fine-tuning the MaxMAC protocol and its properties. Choosing e.g. low values for the thresholds makes sense in delay-sensitive applications, whereas higher values can make sense in energy-sensitive and delay-tolerant applications. We intend to study self-parametrization mechanisms based on estimation of available channel bandwidth, link quality, hopcount, network density in the near future.

4 Simulation Models and Parameters

We implemented the MaxMAC protocol and compared it to S-MAC [5], T-MAC [5], B-MAC [10], WiseMAC [11], X-MAC [6], and the reference protocols IdealMAC and energy-unconstrained CSMA in the OMNeT++ Network Simulator [14]. The IdealMAC protocol has been used in [11] as a reference protocol to show where the *lower bounds* of E^2-MAC protocol efficiency are. IdealMAC models the physical constraints of E^2-MAC protocols, such as the channel bandwidth, the delays and costs of the transceiver switches, as well as the transmission and reception costs. It however assumes that there is no information asymmetry between senders and receivers. Nodes always *know* when they need to switch to receive/transmit in order to handle data transmissions.

In order to reflect the characteristics of wireless propagation (high packet error rate, shadowing and fading-effects), we applied the Log-Normal Shadowing Model [15] implemented in [16]. This channel model allows for a more realistic simulation of wireless channel properties than usual Unit Disk Graph (UDG) based simulation models. It models small-scale shadowing and fading effects -

Table 1. Simulation model parameters

CC1020 [18] parameters		Experiment parameters	
supply voltage V	3 V	simulation runs	100
transmit current I_{tx}	21.9 mA	simulated time	3600 s
recv current I_{rx}	17.6 mA	ARQ max retries	3
sleep current I_{sleep}	1 μA	frame header size	14 bytes
transmission rate R	115.2 $kbps$	payload	50 bytes

which are typical wireless phenomena - for each frame transmission by adding a random perturbation factor to the reception power. The perturbation factor follows a log-normal distribution with a user-selectable deviation σ.

Transceiver and Energy Model: We modeled the state transition delays and the power consumption of wireless sensor nodes using a finite state machine model consisting in the states sleep, receive and transmit, weighted with the respective energy costs. The same methodology is applied in [17], where the power consumption of a IEEE 802.11 wireless device is modeled with the same three states. Experimental results in [17] confirm the adequateness of the linear state transition model. Table 1 lists current, voltage and transmission rate of the CC1020 [18], a byte-level radio transceiver in the 804-940 MHz ISM frequency band. The CC1020 is used by the MSB430 sensor nodes platform [19], which we use for prototyping traffic-adaptive E^2-MAC protocols on real sensor hardware.

E^2-MAC Protocol Simulation Models: Table 2 displays the main parameters of the simulated E^2-MAC protocols. As the protocol behavior often heavily depends on the choice of the essential protocol parameters (e.g. Base Interval, Duty Cycle), we studied the protocols with different *configurations* of those

Table 2. E^2-MAC Protocol Parameters

MaxMAC		B-MAC	
Base Interval	100, 200, 250 ms	Base Interval	25, 50, 100,
Duty Cycle	2, 1, 0.8%		200, 500 ms
LEASE	1 s	Duty Cycle	8, 4, 2, 1, 0.4%
T_1, T_2, T_{CSMA}	4, 8, 12 packets/s	WiseMAC	
S-MAC		Base Interval	25, 50, 100,
Listen Interval	100, 200, 300, 500		200, 500 ms
	1000, 2000 ms	Duty Cycle	8, 4, 2, 1, 0.4%
Duty Cycle	10%	Medium Reservation	u[0,10] $\times$ t_{rx-tx}
T-MAC		X-MAC	
Frame Length	50, 100, 200 ms	Max Interval	200 ms
	300, 500 ms	Min Interval	10 ms
SYNC & RTS size	14 bytes	EarlyACK size	10 bytes
CTS size	10 bytes	CSMA	
SYNC period	10 s	Contention Window	10 ms

parameters, by varying the parameters over a wide range, and not just one particular parameter choice. One such *configuration* would e.g. be B-MAC [Base Interval=200ms, Duty Cycle=1%(2ms)].

For the *slotted* protocols S-MAC and T-MAC, we assume that the nodes' wake-up intervals are synchronized from the beginning of the experiment (as assumed in many MAC studies, e.g. in [11]). With X-MAC, we integrated an adaptation algorithm that adapts the wake/sleep intervals according to incoming packet rate (as specified in [6]), but remains in-between [Max Interval, Min Interval].

WiseMAC implements a cheap collision avoidance using a larger *carrier sensing range* ($\sim 2 \cdot$*hop distance*). Such a mechanism can be accomplished by most of today's radio transceivers by observing the onboard RSSI value and setting appropriate thresholds.

In order to allow for a fair comparison of the E^2-MAC protocol models, we implemented the same packet burst transfer mode for each protocol. Nodes signal pending packets to the receiver and can transmit queued packet trains in bursts, receiving an acknowledgment for each frame.

5 Simulation Results

5.1 Traffic along a Multi-Hop Chain

We simulated a chain consisting of 8 nodes. The source node is generating load, which is then forwarded hop-by-hop towards the sink node, similarly as done in the studies on S-MAC [5] and B-MAC [10]. Almost every existing study on E^2-MAC protocols applies constant rate traffic during each simulation run. In contrast to this, we varied the offered traffic from low rates to high rates during each run, as our major interest is the protocol adaptivity during *run-time*.

Figure 4 displays the offered load generated at the application layer of the source node. The load is low (0.1 packets/s) for most of the time, but there are peaks where the packet rate is increased, up to a maximum rate of 22 packets/s. We chose 22 packets/s as the load maximum as this had proved to be the maximum throughput that CSMA could handle without major packet loss. When increasing the rate above this rate, throughput stalls and additional packets are either dropped due to buffer overflows or are lost due to collisions.

Throughput and Power Consumption: Figure 5 displays the rate of received packets at the sink node vs. simulation time. The curves are averaged from 100 simulation runs for each protocol. As one can clearly see comparing the received packets in Figure 5 with the offered load in Figure 4, IdealMAC manages to handle all packets from source to sink. CSMA only suffers minor packet loss at the load peaks. The throughput of WiseMAC and T-MAC stalls at maximum 8 packets/s and 9 packets/s, respectively, which corresponds to $\sim 35 - 40\%$ of that of CSMA. Figure 5 clearly shows that MaxMAC with its state-based run-time traffic adaptation mechanism reaches the same throughput as energy-unconstrained CSMA. As the protocol adaptively allocates more duty cycles or

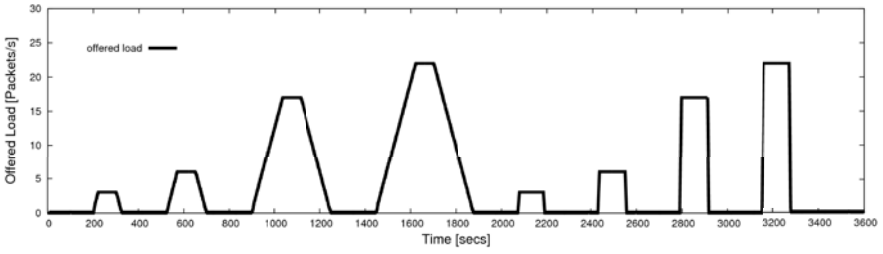


Fig. 4. Offered Load (Packets/s)

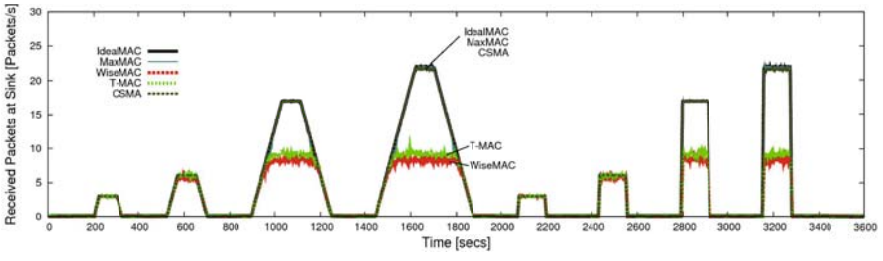


Fig. 5. Throughput at Sink

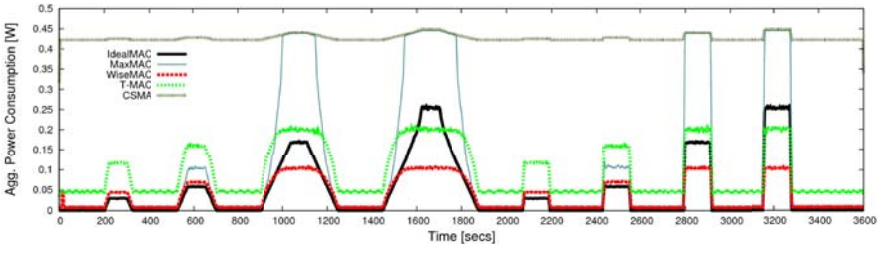


Fig. 6. Aggregated Network Power Consumption

even totally switches to CSMA-like behavior at high traffic rates, the protocol manages to handle the load peaks without major packet loss.

Figure 6 depicts the aggregated power consumption of all 8 sensor nodes' radio interfaces versus simulation time. One can clearly see the big gap between the E^2-MAC protocols and energy-unconstrained CSMA. With low traffic, CSMA wastes a lot of energy on idle listening. The load peaks are hardly visible at all, as the transceiver does not consume much more power when transmitting, compared to idle listening [18]. The IdealMAC reference protocol illustrates the ideal behavior of an E^2-MAC protocol, allocating as much energy as needed to handle the imposed load, and immediately deallocating it with decreasing load. WiseMAC renouncing on costly synchronization schemes has a low per-packet overhead, minimizing preambles by learning adjacent nodes' schedules. It

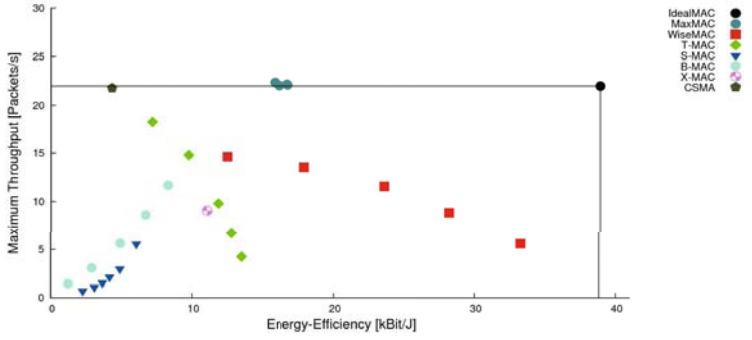


Fig. 7. Throughput vs. Energy-Efficiency

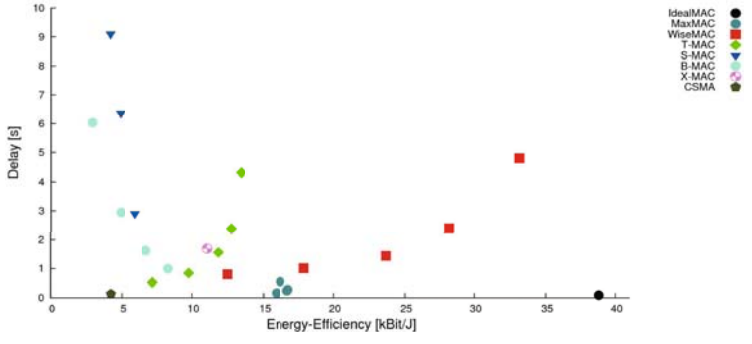


Fig. 8. Delay vs. Energy-Efficiency

exhibits a low power consumption during the low traffic phases, its throughput however stalls at $\sim$ 35% of that of CSMA. T-MAC achieves a slightly higher throughput, but its idle power consumption is above that of WiseMAC, mainly due to the SYNC message overhead to keep the nodes' wake-ups synchronized.

Thanks to the run-time traffic-adaptivity mechanisms of MaxMAC, namely the scheduling of *Extra Wake-Ups*, and the switch to energy-unconstrained CSMA-like behavior with higher traffic load, MaxMAC reaches the same energy-efficiency in the low-traffic-phases as WiseMAC, but is able to handle the load peaks with much lower packet loss. As MaxMAC switches to the CSMA-state with the rate reaching $T_{CSMA} = 12$ packets/s (cf. Table 2), the power consumption of MaxMAC accordingly jumps to the level of CSMA at this rate, too. Figure 6 further illustrates that the *on-demand* resource allocation scheme of MaxMAC further succeeds astonishingly well when the packet rate decreases. With traffic rates decreasing towards 0.1 packets/s after the load peaks, MaxMAC quickly falls back to the states S_2 and S_1 and finally the Base Interval state, where it again exhibits a very low energy-footprint.

Energy-Throughput and Energy-Latency Tradeoffs: E^2-MAC protocols typically trade off quality of service versus higher energy-efficiency. Generally, they introduce higher delays and restrain the maximum achievable throughput. In this subsection we examine the MaxMAC protocol with respect to the energy-throughput and energy-latency tradeoffs and compare it with existing E^2-MAC protocols. Figure 7 and 8 illustrate the measured tradeoffs in the aforementioned experiment. Each dot represents the results of one particular protocol *configuration* in the simulation experiment outlined in Section 4. In Figure 7, the tradeoff between maximum achieved throughput and energy-efficiency of the simulated E^2-MAC protocols becomes well visible. The protocol efficiency is measured in in kbit/J, hence calculating how many *useful* (payload) bits have been transmitted from source to sink for each consumed Joule. A similar concept has been proposed as the energy-per-useful-bit (EPUB) metric in [20] - we however use the reciprocal coefficient in order to obtain a metric where *more is better*. CSMA obviously achieves a high maximum throughput. However, as CSMA never turns off the transceiver, its energy-efficiency remains very low.

IdealMAC illustrates the lower bounds of the E^2-MAC protocol problem in Figures 7 and 8: while it is not possible to reach a higher throughput or a higher efficiency coefficient than IdealMAC, it is neither possible to reach a lower delay. WiseMAC with its short channel polls achieves a high energy-efficiency, especially the configurations with long intervals between two channel polls. The efficiency gain however comes at the cost of a massively restrained maximum throughput and increasing end-to-end latency (cf. Figure 8).

Thanks to its run-time traffic adaptation mechanisms, MaxMAC reaches the same throughput as energy-unconstrained CSMA, but exhibits a much higher energy-efficiency in terms of kbit/J. Although MaxMAC switches to CSMA-like behavior in the high traffic phases, its efficiency coefficient is higher than that of most of today's E^2-MAC protocols. The advantage of achieving the high throughput of CSMA *and* a much better energy-efficiency than most E^2-MAC approaches is a clear novelty in the design space of today's E^2-MAC protocols.

Figure 8 similarly depicts the tradeoff between average packet delay and energy-efficiency. One can observe that CSMA exhibits a very low average delay, however at the cost of a low energy-efficiency. IdealMAC reaches both, a very low delay at a very high energy-efficiency. Thanks to the scheduling of *Extra Wake-Ups*, which reduces the interval between two wake-ups, and the switch to CSMA-like behavior at even higher rates, MaxMAC reaches a far lower average end-to-end latency as other E^2-MAC protocols. MaxMAC achieves a delay which is - given the best examined configuration - only 70% higher than that of CSMA (compared to some 1000% with other E^2-MAC protocols), but achieves an energy-efficiency that is more than three times better than that of of CSMA.

Figure 9 represents the results of each configuration of the simulated E^2-MAC protocols as a tuple in the vector space $X \times Y \times Z$ where X is the energy-efficiency (measured in kbit/J), Y the maximum achievable throughput (packets/s) and Z the average measured delay. The figure illustrates the potential for optimization

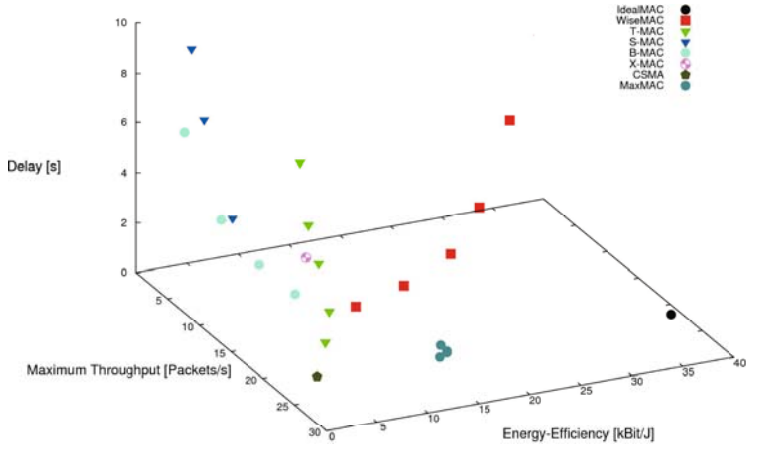


Fig. 9. Energy-Efficiency (x) vs. Maximum Throughput (y) vs. Delay (z)

in the design space of today's E^2-MAC protocols. In [13], we surveyed and compared the adaptivity of the protocols under variable load, using the distance to IdealMAC as a metric to assess the adaptivity of a protocol. [13] concludes that most protocols are not sufficiently adaptive, as they do not alter their behavior with respect to the load conditions. Although there is sufficient channel capacity, most existing protocols still turn their radio transceivers off too aggressively. MaxMAC is clearly distinguishable from the examined reference protocols by its ability to reach the same throughput and a similarly low latency as energy-unconstrained CSMA, while still exhibiting a good energy-efficiency during the considerably long periods of sparse network activity. The three examined *configurations* of MaxMAC hence exhibit the shortest distance to the IdealMAC protocol in the lower right corner in Figure 9, due to the high throughput, low delay and good energy-efficiency measured in the experiment.

5.2 Random Correlated Event Traffic

With our second experiment we examine the behavior of MaxMAC (and the reference protocols) in a larger scenario with a *correlated event workload* model [21]. We simulate a 49-node grid network (7x7) with the center node forming the sink. The distance between two adjacent nodes is 30m. With our parameter settings of the LogNormal channel model [15], packet error rates are $\sim 1\%$ and $\sim 15\%$ on a straight link (30m) and a diagonal link (42.42m), respectively.

We apply a simple event traffic model that mimicks the effects of spacially-correlated events, as proposed in [21] and [22]. Spacially-correlated events are expected to occur in many event-based scenarios for WSNs, e.g. monitoring applications in healthcare [1] systems, disaster-aid systems [2] or tracking applications. The traffic model picks a uniform random (x,y) location for each event. Every node within the event sensing range R of this location then reports data

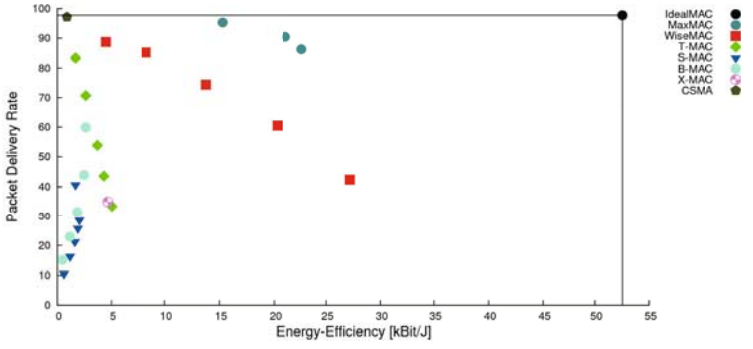


Fig. 10. 49-nodes grid scenario: Packet Delivery Rate (PDR) vs. Energy-Efficiency

packets with a rate of r_{event} during t_{event} towards the sink. We chose values of $R = 30m$, $r_{event} = 6$ packets/s and $t_{event} = 10s$ for the events being triggered each $30s$ at a random location (x,y) of the simulated network. In large event-based scenarios (e.g. a monitoring application), the packet delivery rate (PDR) is usually given higher priority than the throughput per second. We hence measured the packet delivery rate, the average source-to-sink packet delay and the energy-efficiency (in terms of kBit/J) during 100 runs of $3600s$. Packets are routed along the *shortest path*. Nodes select their parent node randomly in the initiation phase of the experiment if there are multiple nodes advertising the same hop count. Energy-efficiency is measured as the total received data bits divided by the aggregated energy spent by all the node's radio interfaces.

Figure 10 depicts the packet delivery rate (PDR) vs. energy efficiency of the different *configurations* of the E^2-MAC protocols in the random correlated event experiment. Energy-unconstrained CSMA and IdealMAC reach a PDR of almost 100%. Some packets are lost due to buffer overflows, as the transmit buffer is assumed to be limited to 10 packets. As CSMA does not turn off the transceiver during the long periods where no traffic occurs, its energy-efficiency remains very low (cf. top-left corner). IdealMAC modeling the *ideal* E^2-MAC protocol behavior reaches the same PDR and a very high efficiency (cf. top-right corner). The configurations of T-MAC and WiseMAC with a short *Base Interval* reach a high PDR, however at the cost of decreasing energy-efficiency. B-MAC and X-MAC reach a modest PDR, but the high per-packet overhead of the B-MAC preambles (which stretch over one entire Base Interval) and the X-MAC preamble strobes negatively impact on their efficiency. Thanks to its run-time adaptation mechanisms, MaxMAC reaches a similar PDR as energy-unconstrained CSMA, while still exhibiting a much higher energy-efficiency. Although the protocol switches to CSMA in the high traffic phases, its overall efficiency is still higher than that of most other E^2-MAC protocols. The combination of a high PDR and a high energy-efficiency achieved by MaxMAC's adaptation mechanisms is well-visible in Figure 10 and constitutes a clear benefit.

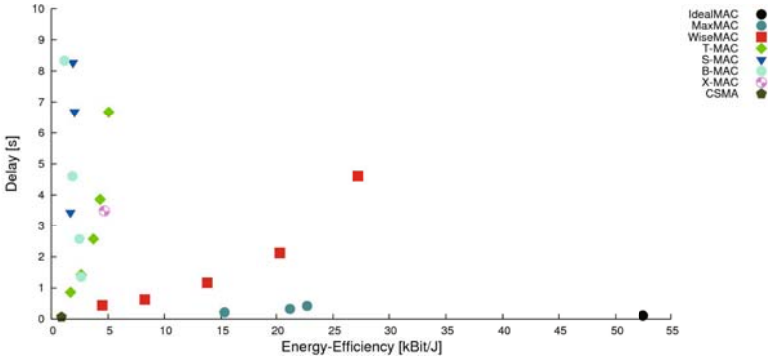


Fig. 11. 49-nodes grid scenario: Delay vs. Energy-Efficiency

Figure 11 depicts the tradeoff between average source-to-sink packet delay and energy efficiency in the random correlated event experiment. CSMA again exhibits a very low average delay at the cost of a very low energy-efficiency, while IdealMAC reaches both, low latency and high energy efficiency. The configurations of T-MAC and WiseMAC with a short *Base Interval* reach a lower average delay, however at the cost of decreasing energy-efficiency. B-MAC and X-MAC have a considerably high delay. As these protocols use long preambles or preamble strobes, latency increases sharply over multiple hops, and sums up to a couple of seconds in the given scenario.

Thanks to the scheduling of *Extra Wake-Ups* and switching to CSMA at higher rates, the three examined *configurations* of MaxMAC reach a far lower average source-to-sink latency as all the other E^2-MAC protocols. The adaptivity concept of MaxMAC further fits to the event-based traffic: with an event being triggered at a random location, nodes start reporting data along the shortest path to the sink. With the load reaching the MaxMAC thresholds T_1, T_2, T_{CSMA}, nodes alter their behavior in order to deliver the pending load. After the event has been processed and the packet stream ends, the LEASE timespans time out and MaxMAC again falls back to the default behavior in the Base Interval state.

A drawback of MaxMAC is the fact that the protocol requires a certain time during which the adaptation mechanisms are triggered. In multi-hop scenarios, all nodes forming a route from the event source to the sink first need to reach the given thresholds. During this *adaptation phase*, packets are lost mainly due to buffer overflows, as the PDR in Figure 10 exhibits. Thereafter the traffic adaptation strategy achieves a high throughput and a low average delay.

6 Conclusions

In this paper we have presented MaxMAC, an E^2-MAC protocol that targets at achieving maximal run-time traffic adaptivity. The protocol targets at event-based sensor network applications where at certain instants, the provision of

high throughput and fast end-to-end response time becomes more important than the conservation of energy. We envision such applications e.g. in healthcare, where nodes attached to patients need to rely on the provision of higher throughput and fast response times when critical values have been sensed, in order to communicate with central entities.

The paper examines MaxMAC in a network simulator and compares it against a selection of other well-known E^2-MAC protocols, an ideal E^2-MAC protocol model and energy-unconstrained CSMA. In both scenarios, MaxMAC is clearly distinguishable from the examined reference protocols by its ability to reach the same throughput and a similarly low latency as energy-unconstrained CSMA, while still exhibiting a good energy-efficiency during long periods of sparse network activity, which are often encountered in event-based monitoring systems. The MaxMAC protocol hence combines the advantages of energy unconstrained CSMA (high throughput, high PDR, low latency) with those of classical E^2-MAC protocols (high energy-efficiency).

References

[1] Malan, D., Fulford-Jones, T., Welsh, M., Moulton, S.: CodeBlue: An ad hoc Sensor Network Infrastructure for Emergency Medical Care. In: MobiSys 2004 Workshop on Applications of Mobile Embedded Systems (2004)
[2] Gao, T., et al.: The Advanced Health and Disaster Aid Network: A Light-weight Wireless Medical System for Triage. IEEE Transactions on Biomedical Circuits and Systems (2007)
[3] Akyildiz, I., Melodia, T., Chowdhury, K.: Wireless Multimedia Sensor Networks: A Survey. Elsevier Computer Networks (2007)
[4] Van Dam, T., Langendoen, K.: An Adaptive Energy Efficient MAC Protocol for Wireless Sensor Networks (TMAC). In: ACM SenSys (2003)
[5] Ye, W., Heidemann, J., Estrin, D.: An Energy Efficient MAC protocol for Wireless Sensor Networks. In: INFOCOM (2002)
[6] Buettner, M., Gary, V.Y., Anderson, E., Han, R.: X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks. In: ACM SenSys (2006)
[7] Lee, S.H., Park, J.H., Choi., L.: AMAC: Traffic-Adaptive Sensor Network MAC Protocol through Variable Duty-Cycle Operations. In: ICC (2007)
[8] Rhee, I., Ajit Warrier, M.A., Min, J.: Z-MAC: a hybrid MAC for Wireless Sensor Networks. In: ACM SenSys (2005)
[9] Ringwald, M., Roemer, K.: BurstMAC: An Efficient MAC Protocol for Correlated Traffic Bursts. In: IEEE Conference on Networked Sensing Systems, INSS (2009)
[10] Polastre, J., Hill, J., Culler, D.: Versatile Low Power Media Access for Wireless Sensor Networks. In: ACM SenSys (2004)
[11] El-Hoiydi, A., Decotignie, J.D.: WiseMAC: An Ultra Low Power MAC Protocol for Multihop Wireless Sensor Networks. In: Nikoletseas, S.E., Rolim, J.D.P. (eds.) ALGOSENSORS 2004. LNCS, vol. 3121, pp. 18–31. Springer, Heidelberg (2004)
[12] El-Hoiydi, A.: Energy Efficient Medium Access Control for Wireless Sensor Networks, PhD Thesis EPFL Lausanne (2005)
[13] Hurni, P., Braun, T.: On the Adaptivity of Today's Energy-Efficient MAC Protocols under varying Traffic Conditions. In: IEEE Conference on Ultra-Modern Technologies, ICUMT (2009)

[14] Varga, A.: The OMNeT++ Discrete Event Simulation System. In: European Simulation Multiconference (2001), http://www.omnetpp.org
[15] Rappaport, T.S.: Wireless Communications: Principles & Practise, 2nd edn. Prentice Hall, Englewood Cliffs (2001)
[16] Kuntz, A., Schmidt-Eisenlohr, F., Graute, O., Hartenstein, H., Zitterbart, M.: Introducing Probabilistic Radio Propagation Models in OMNeT++ MF and Cross Validation Check with NS-2. In: 1st Intl. Workshop on OMNeT++ (2008)
[17] Feeney, L., Nilsson, M.: Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In: INFOCOM (2001)
[18] Texas Instruments CC1020: Single-Chip FSK/OOK CMOS RF Transceiver
[19] Baar, M., Koeppe, E., Liers, A., Schiller, J.: The ScatterWeb MSB-430 Platform for Wireless Sensor Networks. In: SICS Contiki Workshop (2007)
[20] Ammer, J., Rabaey, J.: The Energy-Per-Useful-Bit Metric for Evaluatling and Optimizing Sensor Network Physical Layers. In: SECON (2006)
[21] Hull, B., Jamieson, K., Balakrishnan, H.: Mitigating Congestion in Wireless Sensor Networks. In: ACM SenSys (2004)
[22] Sun, Y., Du, S., Gurewitz, O., Johnson, D.B.: DW-MAC: a Low Latency, Energy Efficient Demand-Wakeup MAC protocol for Wireless Sensor Networks. In: ACM MobiHoc (2008)

Energy-Aware Sparse Approximation Technique (EAST) for Rechargeable Wireless Sensor Networks*

Rajib Rana[1,2], Wen Hu[2], and Chun Tung Chou[1]

[1] School of Computer Sci. and Engineering, University of New South Wales, Australia
{rajibr,ctchou}@cse.unsw.edu.au
[2] CSIRO ICT Center, Australia
wen.hu@csiro.au

Abstract. Due to non-homogeneous spread of sunlight, sensing nodes typically have non-uniform energy profiles in rechargeable Wireless Sensor Networks (WSNs). An energy-aware work load distribution is therefore necessary for good data accuracy while ensuring an energy-neutral operation. Recently proposed signal approximation strategies, in form of Compressive Sensing, assume uniform sampling and thus cannot be deployed to facilitate energy neutral operation in rechargeable WSNs. We propose a *sparse approximation* driven sensing technique (EAST) that adapts sensor node sampling workload according to solar energy availability. To the best of our knowledge, we are the first to propose *sparse approximation* for modeling energy-aware work load distribution in order to improve signal approximation from rechargeable WSNs. Experimental result, by using data from an outdoor WSN deployment, suggests that EAST significantly improves the approximation accuracy while supporting approximately 50% higher *sensor on-time* compared to an approach that assumes uniform energy profile of the nodes.

1 Introduction

Wireless Sensor Networks (WSNs) are currently deployed to monitor micro-climate data from different environments [1, 21]. The Springbrook National Park WSN is one such example. The Springbrook site is part of a World Heritage precinct in Queensland, Australia. CSIRO, in partnership with the Queensland Government Environmental Protection Agency (EPA), is in the process of deploying a WSN of 200 nodes at Springbrook by 2011 to collect micro-climate data for enhancing knowledge of rain forest restoration processes.

Energy supply is a major design constraint in the Springbrook deployment and the lifetime is limited by battery supplies. In the last few years, a large number of research has been conducted ([3] has a comprehensive list) to minimize the radio activities. However, recently it has been reported that many real life applications require specific sensors whose power consumption is significant [17].

* This work was done while Rajib Rana was an intern at CSIRO ICT center, Australia.

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 306–321, 2010.

Table 1. Energy consumption of some common radios [15, 21]. T_x and R_x are the transmission and the reception energy accordingly. We compute transmission energy for a 32 byte data packet.

Radio	Producer	Energy Consumption
CC2420	Texas Instruments	T_x:34 μJ R_x:38 μJ
CC1000	Texas Instruments	T_x:40 μJ R_x:28 μJ

Table 2. Energy Consumption of some common sensors [21]. Sensors are turned on for 5 seconds for one reading (Sensors are turned on for 5 seconds every 5 minutes in the Springbrook deployment.).

Sensor	Sensing	Energy Consumption
Met One 034B	Wind Speed	45 mJ
Met One 034B	Wind Direction	45 mJ

In addition, longer acquisition times of some specific sensors may even result in significantly higher energy consumptions than the radio (see Table 1 and 2 for a comparison of energy consumptions of some popular radio equipment with the energy hungry wind sensors). In order to cope with the increasing energy demand, a number of sensor deployments are adopting a complementary approach of supplementing the energy supply of the system by harvesting additional energy from the environment [21, 11].

Out of the variety of energy harvesting modalities, solar current harvesting provides one of the highest power densities [18]. However, solar energy will typically not be homogeneously spread over the network which results in non-homogeneous energy profile (i.e non-uniform solar current harvest rates) of the sensing nodes. Therefore, sensing task allocation that assumes uniform energy profile of the sensing nodes could deplete the energy of a number of nodes and create holes in the network connectivity or coverage. In order to avoid such situation, the Springbrook deployment reduces the fraction of time the sensors are turned on to take samples (we refer this quantity as *sensor on-time*) to less than 2% for all nodes, which results in poor approximation of the signal.

Data collected from the wireless sensor deployments are typically correlated and therefore compressible [4] in an appropriate transform. Recent results in Compressive Sensing [6] suggests that if the data is compressible, a signal vector with $\hat{N}$ data values can be well approximated using only $k(<< \hat{N})$ transform coefficients. If the k largest coefficients could be approximated from a small number of measurements, where measurements are taken with high probability from energy-rich sensing nodes and with smaller probability from energy-constrained nodes, we could approximate the signal with good accuracy while ensuring an energy neutral operation. An energy neutral operation means that the energy consumption should be less than the energy harvested from the environment. The estimation techniques of compressive sensing ([20, 4, 9]) have so far assumed that the signal is sampled uniformly. Therefore, in order to approximate a signal with good accuracy while ensuring an energy neutral operation, a theoretical framework that supports nonuniform sampling need to be developed. In this paper we address this challenge. Our contributions are as follows

1. We present a distributed sensing framework, EAST, which for the first time implements *sparse random projections* to distribute sensing workload based on the solar energy harvest rates of the nodes to achieve an energy-neutral operation while at the same time is able to approximate a signal with good accuracy with high probability. Our work therefore draws a connection between compressive sensing and the sensor selection problem.
2. We determine the upper bound of sampling requirement of EAST as a function of g_j, which is a parameter that determines the sparsity of projection matrix and is proportional to the energy harvest rate of the sensing node n_j, and show that $O(poly(k, \log \hat{N})\Sigma_{j=1}^{N} \frac{1}{g_j})$ sparse random projections are sufficient for EAST to reconstruct a signal with error, comparable to the best k-term approximation.
3. We evaluate EAST using the data collected from the Springbrook sensor deployment and report that energy-aware task distribution allows EAST to support approximately 50% higher sensor on-time, and thus allows EAST to achieve significantly better approximation compared to a sensing technique that assumes uniform energy profile of the nodes. Experimental result also reveals that EAST can achive approximation accuracy close to the best k-term approximation.

The remainder of the paper is organized as follows. In the next section, we precisely define EAST and describe the necessary modeling assumptions. Then we model EAST in Section 3 and describe a distributed algorithm for EAST in Section 4. We provide the evaluation result in Section 5 and discuss the related literature in Section 6. Finally, we conclude in Section 7.

2 Problem Definition

Consider a signal x captured over time t_h, $1 \leq h \leq M$ from N nodes n_j, $1 \leq j \leq N$ of a WSN. Assume that the network is rechargeable using solar energy. Define E^j be the amount of energy harvested by node n_j during time t_h, $1 \leq h \leq M$(in the rest of the paper we refer to E^j as the energy profile of the node). Due to non-uniform spread of sunlight, E^j can be non-uniform, e.g., nodes in the open space can have higher E^j whereas nodes in the forest can have smaller E^j. We want to develop a sensing framework that distributes sampling workload based on E^j (precisely we want the energy-rich sensors to work more and thus reduce the work load of energy-constrained sensors) and at the same time minimizes the approximation error while ensuring an energy-neutral operation.

Let us further define an indicator variable

$$f_{hj} = \begin{cases} 1, & \text{if sensor } n_j \text{ is turned on at } t_h \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

In order to ensure an energy neutral operation, we turn on sensor n_j[1] (i.e., f_{hj} will be 1) based on its energy profile E^j. Consequently, some of the values of f_{hj} could be zero. Note that the value of the signal x at time instances where $f_{hj} = 0$ are not measured, therefore, we need a method to compute an approximation of those components in x that have not been measured. We aim to develop a method that achieves good approximation while maintaining energy-neutral operation.

In order to simplify the description, we will assume $M = 1$ for the rest of this Section as well as in Section 3. This means that x is a 1-dimensional vector and the j-th component of x is in fact the sensor measurement of sensor n_j.

2.1 Compressible Data

Data collected from the wireless sensor deployments are typically correlated and therefore compressible in an appropriate transform [4]. Let us consider a transform $\Psi \in \mathbb{R}^{N \times N}$ (Wavelets or Discrete Fourier Transform are typically used as transforms), consisting of a set of orthonormal basis vectors $\{\psi_1..\psi_N\}$. A signal x is compressible, if the reordered transform coefficients $\theta = [\psi_1^T x, .., \psi_N^T x]^T$ decay like power law [6], i.e., the π-th largest transform coefficient satisfies

$$|\theta|_{(\pi)} \leq R\pi^{-\frac{1}{s}} \tag{2}$$

for each $1 \leq \pi \leq N$, where R is a constant, and $0 \leq s \leq 1$. We will call s the compressibility parameter.

Recent results [6] of compressive sensing show that if the data is compressible, the largest (in magnitude) k transform coefficients (θ) capture most of the signal information. A compressible signal can therefore be well approximated by recovering only the k largest transform coefficients. The approximation that keeps the k largest transform coefficients and discards the remaining as zero is called the *best k-term approximation* [20]. In order to model EAST we assume that the data collected at the energy-constrained nodes are correlated to the data collected at the energy-rich nodes and thus, if we collect large amount of data from the energy-rich nodes (and a small amount of data from energy constrained nodes), the k-largest coefficients could be recovered to have a good approximation of the signal.

2.2 Sparse Random Projections

In the literature it has been shown that if the signal is compressible, ℓ sparse random projections can be used to recover the signal with approximation error comparable to the best k-term approximation with high probabilities [20]. Unlike dense projection matrix (typically used in Compressive Sensing), the degree

[1] Note that, in WSN literature, a sensor can be used to refer to a *sensor node* (which includes a CPU, a radio and measurement sensors) or a *measurement sensor* (e.g. a temperature sensor, a wind speed sensor). In this paper, we refer to *turning on sensor* n_j as to turning on the measurement sensor on node n_j.

of sparsity of the sparse random projections can control the number of measurements need to be acquired. For example, consider a sparse projection matrix $\Phi \in \mathbb{R}^{\ell \times N}$ with following entries.

$$\Phi_{ij} = \sqrt{\rho} \begin{cases} +1 & \text{with prob. } \frac{1}{2\rho} \\ 0 & \text{with prob. } 1 - \frac{1}{\rho} \\ -1 & \text{with prob. } \frac{1}{2\rho} \end{cases}$$

ρ determines the sparsity of the random projections. Thus, if $\frac{1}{\rho} = 1$, the random matrix has no sparsity (i.e., it is dense); on the other hand, if $\frac{1}{\rho} = \frac{1}{N}$, the matrix is sparse and the expected number of non-zero elements in each row of the projection matrix is $1(= N/N)$.

In order to see how sparse projections can reduce the sampling requirement, let us first point out that the vector u $(= \Phi x)$ is required for signal estimation. Note that if for a particular value j, we have $\Phi_{ij} = 0$ for all i (in other words, the j-th column of Φ is all zero), then the j-th component of x is not needed to obtain u. This means that node n_j does not need to turn its sensor on to collect a sample. For the Φ_{ij} defined above, the mean number of sensors that are required to sample is given by $N(1-(1-\frac{1}{\rho})^{\ell})$, which can be showed to be bounded from above by $\frac{N\ell}{\rho}$. For $\rho = N$, this means at most ℓ samples are required. Since ℓ is supposed to be significantly less than N, the sampling requirement is low.

3 Modeling EAST

We use sparse random projections to model EAST. We control the sparsity of the projection matrix based on the energy profile such that measurements from energy rich sensing nodes are taken with high probabilities and those are taken from the energy constrained sensing nodes with small probabilities. Our sparse projection matrix $\Phi \in \mathbb{R}^{\ell \times N}$ has the following entries

$$\Phi_{ij} = \sqrt{\frac{1}{g_j}} \begin{cases} +1 & \text{with prob. } \frac{g_j}{2} \\ 0 & \text{with prob. } 1 - g_j \\ -1 & \text{with prob. } \frac{g_j}{2}. \end{cases} \tag{3}$$

Here $g_j = \frac{E^j}{\Sigma_{j=1}^{N} E^j} * \frac{\ell}{N}$ gives the probability of a measurement from sensor n_j to be included in the i-th projection. Note that g_j is proportional to the energy profile of node n_j, therefore higher energy profile of a node will increase the probability of inclusion of measurement from the node. If $\Phi_{ij} \neq 0$, we want measurement from sensor n_j to be included in the i-th projection. In order to control sensor scheduling based on Φ_{ij}, we determine the value of the indicator variable f_{hj} based on the values of Φ_{ij}. Let us consider one time snapshot (t_1) of the data $(x_1..x_N)$. Sensor scheduling to acquire this snapshot is determined by

$$f_{1j} = \begin{cases} 0 & \text{if } \Sigma_{i=1}^{\ell} |\Phi_{ij}| = 0 \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

We now prove that if the signal satisfies peak-to-total energy condition,

$$\frac{||x||_\infty}{||x||_2} \leq \mu \tag{5}$$

EAST approximates the signal with error comparable to the best k-term approximation with high probability. Note that the peak-to-total energy condition (5) can be related to the signal compressibility. If signal x is compressible in a transform with compressibility parameter s, then the peak-to-total energy [20],

$$\frac{||x||_\infty}{||x||_2} \leq \mu = \begin{cases} O(\frac{logN}{\sqrt{(N)}}) & \text{if } s = 1 \\ O(\frac{1}{\sqrt{(N)}}) & \text{if } 0 < s < 1. \end{cases} \tag{6}$$

We prove that EAST can approximate a signal with error comparable to the best k-term approximation in two stages. In the first stage (Proposition 1), we show that sparse random projections can produce estimation for the transform coefficients of the data. Then in the second stage (Proposition 2), we show that the approximation error of the estimation is comparable to the best k-term approximation.

Note that the transform coefficients of the data are the inner product between the data and the set of orthonormal bases. Therefore, we first show that sparse random projections of our projection matrix preserve inner products within a small error, with high probability. Proposition 1 states that an estimation of the inner product between two vectors, using only the random projections defined by Equation (3), has the correct expectation with bounded variance (The proof of Proposition 1 is shown in Appendix).

Proposition 1. *Let Φ be the projection matrix given by Equation (3). Define $u = \frac{1}{\sqrt{\ell}}\Phi x$ and $v = \frac{1}{\sqrt{\ell}}\Phi y \in \mathbb{R}^\ell$ as the random projection of two vectors x and $y \in \mathbb{R}^\ell$. Expectation and variance of the inner product of u and v are respectively*

$$\mathbb{E}\left[u^T v\right] = x^T y \qquad \text{and}$$

$$Var\left(u^T v\right) = \frac{1}{\ell}\left(\left(x^T y\right)^2 + ||x||_2^2||y||_2^2 + \Sigma_{j=1}^N \frac{1}{g_j} x_j^2 y_j^2 - 3\Sigma_{j=1}^N x_j^2 y_j^2\right).$$

It can be observed that the variance of the estimation is largely controlled by the factor $\Sigma_{j=1}^N \frac{1}{g_j}$. Thus, if g_j is a small value for a node n_j, the estimation will have high variance. Note that g_j is proportional to the energy profile E^j, therefore when all the nodes have good access to sunlight, good estimation can be produced. Apart from $\Sigma_{j=1}^N \frac{1}{g_j}$, the variance of the estimation is also significantly controlled by the number of projections. A large value of ℓ could produce a smaller variance. In Proposition 2, we will determine the value of ℓ based on the factor $\Sigma_{j=1}^N \frac{1}{g_j}$. Note that in [20] it is shown that the variance of this estimation is controlled by the number of projections (ℓ) only and it is not shown that how the variance will be changed if the nodes have non-uniform energy profile.

Having showed that the estimation of the inner product between two vectors, using only the sparse projections of those vectors, has a good quality estimation with bounded variance, it can be shown (see Lemma 1 in Appendix) that the error of the estimation $\hat{a}_i$ for $x^T y_i$, using the sparse random projections $\frac{1}{\sqrt{\ell}}\Phi x$ and $\frac{1}{\sqrt{\ell}}\Phi y_i$, satisfies

$$|\hat{a_i} - x^T y_i| \leq \epsilon ||x||_2 ||y_i||_2, \forall_{i=1..N}. \tag{7}$$

Finally, in Proposition 2 we state that the estimation error determined in Equation (7) is comparable to the k-term approximation with high probability (proof is included in Appendix).

Proposition 2. *Assume data $x \in \mathbb{R}^N$ satisfies the peak-to-total energy condition (5), and with*

$$\ell = O(\frac{1+\gamma}{\epsilon^2 \eta^2} k^2 \mu^2 \log N \Sigma_{j=1}^{N} \frac{1}{g_j})$$

the sparse random matrix $\Phi \in \mathbb{R}^{\ell \times N}$ satisfies condition (9). Denote $u = \frac{1}{\sqrt{\ell}}\Phi x$ as the sparse random projection of x and $\Psi \in \mathbb{R}^{N \times N}$ as an orthonormal transform. Transform coefficients of x in Ψ is given by, $\theta = \Psi^{-1} x$. Assume the best k-term approximation gives an approximation ($\hat{x}_{opt}$) with error $||x - \hat{x}_{opt}||_2^2 \leq \eta ||x||_2^2$. Using only u, Φ and Ψ, x can be recovered with error

$$\frac{||x - \hat{x}||_2^2}{||x||_2^2} \leq (1+\epsilon)\eta \tag{8}$$

with probability at least $1 - N^{-\gamma}$.

From Proposition 2 it can be observed that a smaller value of the peak-to-total energy (μ) makes the requirement of number of projections ℓ to be small (this is inherent to sparse approximation). However, ℓ is largely controlled by the factor $\Sigma_{j=1}^{N} \frac{1}{g_j}$. Thus, if g_j is a small value for a node n_j, a large number of projections is required to achieve an accuracy similar to the best k-term approximation. One of the main contributions of our work is that we enable energy-aware work load allocation and thus support a large ℓ to achieve an accuracy comparable to the best k-term approximation.

4 Distributed Algorithm

Energy-aware work load allocation typically increases the amount of communications between node and the base station and thus increases the consumption of transmission energy. We therefore design a distributed algorithm for EAST, which generates projections with a few communications between the sensing nodes and the base station.

Note that our description so far has assumed $M = 1$, however the framework can be readily extended to the case with $M > 1$. In this case, we consider the

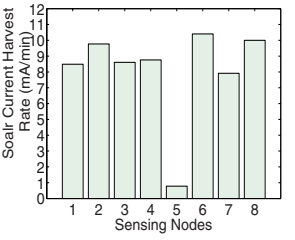
sensor measurement x_{hj} collected at time t_h $(h = 1, ..., M)$ by sensor n_j $(j = 1, .., N)$. Since the algorithm in Section 3 works with a vector, we will *vectorize* the 2-dimensional signal x_{hj}. We will abuse the notation and use x to denote this vector (this should be clear from the context). The vector x has $\hat{N} = MN$ elements where the q-th element of x is x_{hj} where $q = h + (j-1) * M$. The corresponding projection matrix Φ is now an $\ell \times \hat{N}$ matrix. For $q = h+(j-1)*M$, the elements in the q-th column of the projection matrix (Φ_{iq} with $i = 1, ..., \ell$) are generated by Equation (3) with parameter g_j and these elements will determine whether the sensor n_j will sample at time t_h. We will now describe an algorithm which is used by EAST to recover an approximation of the signal (x), from the sparse projections created locally in different nodes.

- Initially, sensor node $n_{\tilde{j}}$ $(1 \leq \tilde{j} \leq N)$ locally decides to generate $\ell_{\tilde{j}}$ rows of the projection matrix where $0 \leq \ell_{\tilde{j}} \leq \ell$ and $\sum_{\tilde{j}=1}^{N} \ell_{\tilde{j}} = \ell$.
- Then each node $n_{\tilde{j}}$ $(1 \leq \tilde{j} \leq N)$ generates the random numbers $\Phi_{r1}, ..., \Phi_{r\hat{N}}$ using the distribution function mentioned in Equation (3)(We assume that node $n_{\tilde{j}}$ is responsible for generating the r-th row $(1 \leq r \leq \ell)$ of the projection matrix. Consider the element Φ_{rq} in the projection matrix and let us assume that the column index q and the node-time pair (j, h) have one-to-one correspondence given by $q = (j-1) * M + h$.).
- If $\Phi_{rq} \neq 0$, node $n_{\tilde{j}}$ tasks node n_j to sample at time t_h and node n_j sends the sample to node $n_{\tilde{j}}$.
- Upon receiving x_{jh} from node n_j, $n_{\tilde{j}}$ computes $u_r = \Sigma_{q=1}^{\hat{N}} \Phi_{rq} x_q$ (where $x_q = x_{jh}$). Node $n_{\tilde{j}}$ performs this operation for all the values it receives and finally transmits u_r to the base station. This process is repeated for all node $n_{\tilde{j}}, 1 \leq \tilde{j} \leq N$.
- After receiving transmissions from the nodes, base station has $\Phi_{\ell \times \hat{N}} x = [u_1, ..., u_\ell]^T$. It then generates $\Phi_{\ell \times \hat{N}}$ using the same seed as the nodes. Finally, with $u(= \Phi_{\ell \times \hat{N}} x)$, $\Phi_{\ell \times \hat{N}}$ and Ψ, base station uses *AMS sketching decoder* [2] to recover the signal.

5 Evaluation

In this Section we evaluate the performance of EAST using energy hungry wind speed and wind direction sensor data, collected from the Springbrook sensor deployment. We have used data from 8 of the sensing nodes at the Springbrook deployment, where among these 8 nodes (shown in Fig. 1(a)), node 5 is deep in the forest whereas the rest of the nodes are in the open space. Consequently, solar current harvest rate of node 5 is the lowest whereas the rest of the nodes have higher and also similar harvest rates (see Fig. 1(b)). Inter-sampling interval in the deployment is 5 minutes and we collected 1 month data which gives us 8640 snapshots for both of the wind sensor data. As *AMS sketching decoder* computes the estimation from median, it performs better with large $\hat{N}$. Therefore, we have arbitrarily chosen large $\hat{N}$=2048. We made 30 smaller datasets from 8640 (we

(a) Positions of the sensing nodes at Springbrook deployment. Node 5 is in the deep forest whereas rest of the nodes are mostly in the open space.



(b) Energy profile of the sensing nodes at Springbrook deployment

Fig. 1. Location and energy harvesting rate of the Springbrook sensing nodes

used 7680 out of these 8640 snapshots and thus discarded the last few snapshots) snapshots, where each set has $M = 256$ snapshots from $N = 8$ nodes. Thus, we get $\hat{N} = MN$ =2048. Note that we also verified that for other large values of $\hat{N}$, such as 512 and 1024, EAST produces similar approximation.

5.1 Uniform-Energy Sensing Technique, UEST

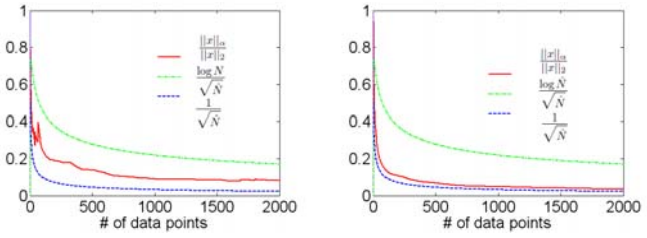
We compare the performance of EAST with a uniform energy sensing technique (UEST). UEST assumes that nodes have homogeneous energy profile and therefore allocates sampling workload uniform randomly. In particular we use EAST to create UEST where we deliberately modify the energy profile E^j of all sensing nodes $1 \leq j \leq N$ to be equal (We use equal energy profile $\hat{E}^j = 1/8\Sigma_{1 \leq j \leq N} E^j$). In addition to providing a way to compare EAST with a sensing technique which assumes uniform energy in all sensing nodes, UEST also facilities the evaluation of EAST at uniform energy condition.

5.2 Approximation Error

Let $\hat{x}$ be the approximation of the signal x, we use relative error, $||x-\hat{x}||_2^2/||x||_2^2$ to determine the accuracy of the approximation. The relative error is a commonly used error metric in the signal processing literature [12, 20] that tells us how close the approximate signal is to the real signal.

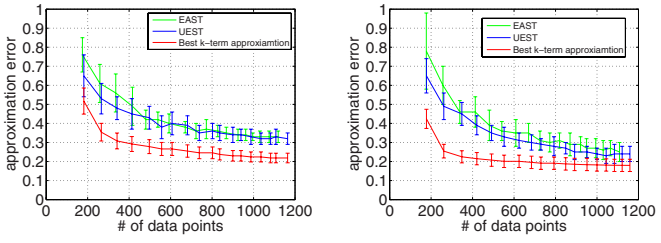
5.3 Results

Peak-to-total energy condition is a sufficient condition for sparse approximation. In Fig. 2 we find that for both of the wind data $\frac{||x||_\alpha}{||x||_2}$ is bounded by $\frac{\log \hat{N}}{\sqrt{\hat{N}}}$ and $\frac{1}{\sqrt{\hat{N}}}$. Therefore, according to Equation (6) the wind sensor data obey the peak-to-total energy condition. From Fig. 2 we also observe that the compressibility



(a) Wind Speed (b) Wind Direction

Fig. 2. Peak-to-total energy condition on data

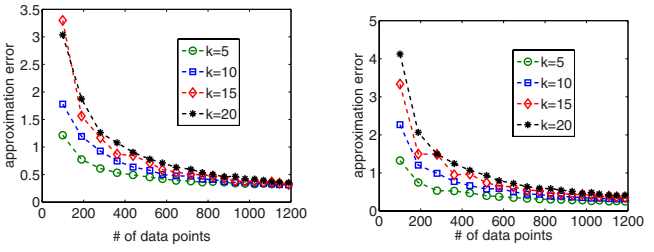


(a) Wind Speed (b) Wind Direction

Fig. 3. A comparison of the approximation error of the micro-climate data using EAST, UEST, and optimal Haar wavelet based approximation. The relative approximation error is plotted versus the number of random projections $\ell = k^2 \log \hat{N}$ for $\hat{N}$=2048. The error bars show the standard deviation of the approximation error.

parameter s is bounded by $0 < s \leq 1$, therefore, the wind sensnor data are also compressible (see Equation (2)).

Fig. 3 compares the approximation accuracy of EAST and UEST against different number of data points. We vary the number of projections (ℓ) and extract the number of data points included in the ℓ projections and then plot the approximation accuracy against the number of data points to precisely demonstrate the sensing requirements of EAST. For each number of data points, we use the snapshots collected from the Springbrook deployment to compute the mean and standard deviation of the approximation error. We observe that unless for very small number of data points, the mean and the standard deviation of the approximation error using EAST is as good as UEST. Precisely, by using 400(= 19%) data points, EAST achieves an approximation error below 0.5. Fig. 3 also compares the approximation of EAST and UEST with the best k-term approximation. We observe that both EAST and UEST performs closer to the best k-term approximation when the number of data points are more than 800.



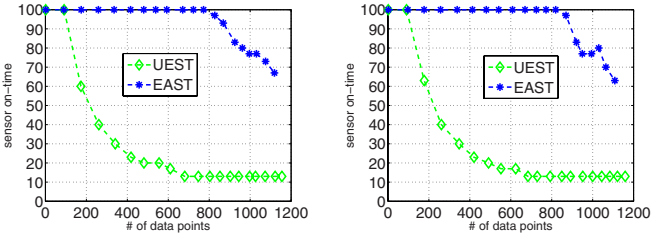
(a) Wind Speed (b) Wind Direction

Fig. 4. Impact of the number of significant coefficients retained (k) on the approximation of EAST. The relative approximation error is plotted versus the number of sampled data points for different value of k. Reconstruction is poor for $k = 1$ and thus is excluded from the figure.

Using sparse approximation, we only reconstruct the significant coefficients of a signal in the transform domain and let the insignificant coefficients to be zero. We have investigated the impact of the number of significant coefficients being retained (k) on the accuracy of approximation and found that, unless using very small value (e.g. $k = 1$), accuracy of the approximation is similar for different values of k. However, in order to avoid cluttering the images, in Fig. 4, we plot $\frac{||x-\hat{x}||_2^2}{||x||_2^2}$ versus k for k up to 20. We use $k = 5$ for the rest of the paper.

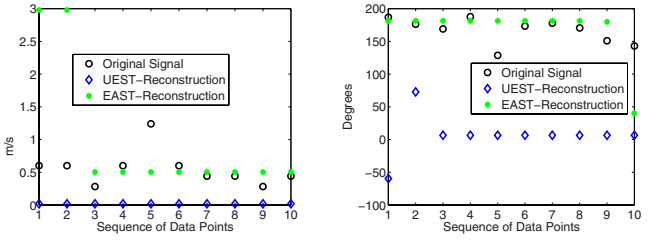
One of the major contributions of this paper is, EAST attempts to minimize approximation error by increasing sensor on-time. Here we use the term *sensor on-time* to indicate the fraction of time a sensor is on when it takes a sample (For example, for an inter-sampling period of 5 minutes and a sensor on-time of 0.6, the sensor will be turned on for 0.6*5 = 3 minutes every time the sensor takes a sample. Note that it can be shown that the duty cycle of a sensor is given by the product of its sampling probability and sensor on-time.). The sensor on-time is common for the network, however, the sampling probability of a sensor is determined by its energy profile. In order to show that EAST supports longer sensor on-time, we compare the maximum sensor on-time (while maintaining an energy neutral operation) supported by EAST and UEST for different number of data points. Note that in the Springbrook deployment, battery voltage $V = 3$ Volts, the electrical current used to acquire a wind sample is $I = 3$ mA and the inter-sampling interval is $T = 5$ minutes (300 seconds). Therefore, if the sensor on-time is ω, then the amount of energy spent by sensor n_j over the period t_h (where $h = 1, ..., M$) is given by $\Delta_j = VI\omega T\Sigma_{1\leq h\leq M} f_{hj}$. The maximum sensor on-time that is supported by the network is the maximum value of ω such that $\Delta_j \leq E^j$ for all j.

In Fig. 5 we compare supported sensor on-time for different number of data points. It is observed that, when we use 1200($< 50\%$) data points, EAST can support 50% longer sensor on-time compared to UEST.



(a) Wind Speed (b) Wind Direction

Fig. 5. Comparison of sensor on-time supported by EAST and UEST at energy neutral condition. The relative approximation error of the data is plotted versus the number of sampled data points for $\tilde{N}$=2048.



(a) Wind Speed (b) Wind Direction

Fig. 6. (a) Recovery of temporal signal at node 5. In order to avoid cluttering of image we show only 10 data points of the signal. Similar recovery is observed for rest of the data points.

Now let us show the impact of sensor on-time on the accuracy of approximation. In Fig. 6 we plot the approximated signal along with the real signal collected in node 5. We choose node 5 deliberately to show the robustness of EAST. Note that node 5 has the lowest energy profile and therefore has the least sampling probability. We use 1200 data points and sensor on-time to be 0.5 for this approximation. While using UEST, node 5 fails due to exceeding its energy budget, which causes poor approximation, whereas energy-aware workload distribution yields significantly better approximation for EAST.

6 Related Work

A large number of signal approximation techniques use Compressive Sensing [7, 20, 4, 9] to conserve transmission energy assuming that radio is the dominant component of energy consumption, however we assume energy-hungry sensor dominates the energy consumption.

In [5] an adaptive sampling algorithm is presented which can be used for estimating the best sampling frequency for energy hungry sensors. However, similar to the work of compressive sensing [20, 4, 9] their approach assume that the sensors have uniform energy profiles.

Work presented in [13] proposes a harvest-aware adaptive sampling approach to dynamically identify the maximum duty cycle. However, their focus is not on signal approximation from the network.

An application-specific approach for energy conservation is presented in [23] where adaptive sampling and energy-aware routing are applied jointly to recover a signal. However, we consider energy-aware data acquisition in our paper.

In [19], a Bayesian estimation technique is presented to estimate the wind speed and wind direction signals. They have supplemented their estimation using the assumption that the wind speed and wind direction signals have a correlation with hourly tide data. However, in our work we assume that signals are compressible due to the presence of spatial-temporal correlation among the data collected at different sensing nodes.

A number of studies [14, 22, 10, 8] have proposed to exploit the spatial-temporal correlation of the signal to reduce sampling requirements. Though our approach has similar assumption, we have considered non-uniform energy profile of the sensors which is different. Moreover, we have used *Sparse Approximation* which is also different from their approaches.

A Compressive Sensing based data gathering approach is presented in [16] which investigates the impact of a routing topology generated sparse projection matrix on the accuracy of the approximation. Our work is different from theirs since our projection matrix is not based on the routing topology rather it is populated based on the energy profile of the sensors.

7 Conclusion

This paper proposes an energy-aware sensing technique (called EAST) that implements distributed sparse random projections to adapt sampling workload distribution based on the solar energy availability at nodes, and thus recovers an approximation of the signal with good accuracy while ensuring an energy neutral operation. A large number of recently developed compressive sensing driven approximation strategies assume that each element of the projection vector is drawn from the same probability distribution. This inherently assumes uniform sampling and thus is inapplicable for ensuring energy neutral operation when nodes have non-uniform energy profiles.. We develop a theoretical framework to determine the number of projections need to be collected as a function of the energy profile of the nodes and prove that $O(poly(k, \hat{N})\Sigma_{j=1}^{\hat{N}} \frac{1}{g_j})$ sparse projections are sufficient to reconstruct a $\hat{N}$ data point signal with accuracy comparable to the best k-term approximation. We apply EAST to reduce the energy consumption of wind speed and wind direction sensors; however, EAST is general and can be used for any signal that satisfies the peak-to-total energy condition. Evaluation result shows that EAST increases the sensor on-time by approximately

50% and thus offers significantly better approximation of a signal compared to a sensing technique that assumes uniform energy profile of nodes. Experimental result also supports that approximation accuracy of EAST is close to the best k-term approximation.

References

[1] Habitat monitoring on great duck island, http://www.greatduckisland.net/index.php

[2] Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. Journal of Computer and System Sciences, 20–29 (1996)

[3] Anastasi, G., Conti, M., Francesco, M., Passarella, A.: Energy conservation in wireless sensor networks: A survey. Ad Hoc Networks 7(3), 537–568 (2009)

[4] Bajwa, W., Haupt, J., Sayeed, A., Nowak, R.: Compressive wireless sensing. In: IPSN 2006, 134–142 (2006)

[5] Alippi, C., Anastasi, G., Francesco, M.D., Roveri, M.: An adaptive sampling algorithm for effective energy management in wireless sensor networks with energy-hungry sensors. In: IEEE-Transactions on Instrumentation and Measurement

[6] Candés, E.: Compressive sampling. In: Proc. of the Int. Congress of Mathematics (2006)

[7] Chou, C.T., Rana, R., Hu, W.: Energy efficient information collection in wireless sensor networks using adaptive compressive sensing. In: Proc. of the 34th Annual IEEE Conference on Local Computer Networks (LCN 2009), pp. 443–450 (2009)

[8] Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J., Hong, W.: Model-driven data acquisition in sensor networks. In: VLDB 2004, pp. 588–599. VLDB Endowment (2004)

[9] Duarte, M.F., Wakin, M.B., Baron, D., Baraniuk, R.G.: Universal distributed sensing via random projections. In: IPSN 2006, pp. 177–185 (2006)

[10] Gupta, H., Navda, V., Das, S., Chowdhary, V.: Efficient gathering of correlated data in sensor networks. ACM Trans. Sen. Netw. 4(1), 1–31 (2008)

[11] Hu, W., Bulusu, N., Chou, C.T., Jha, S., Taylor, A., Nghia, V.: Design and evaluation of a hybrid sensor network for cane toad monitoring. ACM Trans. Sen. Netw. 5(1), 1–28 (2009)

[12] Ji, S., Xue, Y., Carin, L.: Bayesian compressive sensing. IEEE Trans. Signal Processing (2007)

[13] Kansal, A., Hsu, J., Zahedi, S., Srivastava, M.B.: Power management in energy harvesting sensor networks. Trans. on Embedded Computing Sys. 6(4), 32 (2007)

[14] 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)

[15] Polastre, J.R.: PhD thesis

[16] Quer, G., Masiero, R., Munaretto., D., Rossi, M., Widmer, J., Zorzi, M.: On the interplay between routing and signal representation for compressive sensing in wireless sensor networks. In: ITA (2007)

[17] Raghunathan, V., Ganeriwal, S., Srivastava, M.: Emerging techniques for long lived wireless sensor networks. IEEE Communications Magazine 44(4), 108–114 (2006)

[18] Raghunathan, V., Kansal, A., Hsu, J., Friedman, J., Srivastava, M.B.: Design considerations for solar energy harvesting wireless embedded systems. In: IPSN 2005, p. 64 (2005)

[19] Reboul, S., Benjelloun, M.: Joint segmentation of the wind speed and direction. Signal Process. 86(4), 744–759 (2006)
[20] Wang, W., Garofalakis, M., Ramchandran, K.: Distributed sparse random projections for refinable approximation. In: IPSN 2007, pp. 331–339 (2007)
[21] Wark, T., Hu, W., Corke, P., Hodge, J., Keto, A., Mackey, B., Foley, G., Sikka, P., Brunig, M.: Springbrook: Challenges in developing a long-term rainforest wireless sensor network. In: ISSNIP (December 2008)
[22] Willett, R., Martin, A., Nowak, R.: Backcasting: adaptive sampling for sensor networks. In: IPSN 2004, pp. 124–133 (2004)
[23] Zhou, J., De Roure, D.: Floodnet: Coupling adaptive sampling with energy aware routing in a flood warning system. J. Comput. Sci. Technol. 22(1), 121–130 (2007)

Appendix

Proof (Proof of Proposition 1). It can be proved that the projection matrix defined by Equation (3) satisfies these conditions:

$$\mathbb{E}\left[\Phi_{ij}\right] = 0, \mathbb{E}\left[\Phi_{ij}^2\right] = 1, \mathbb{E}\left[\Phi_{ij}^4\right] = \frac{1}{g_j}. \tag{9}$$

Define independent random variables $w_1, ..w_\ell$ where, $w_i = \left(\Sigma_{j=1}^N x_j \Phi_{ij}\right)\left(\Sigma_{j=1}^N y_j \Phi_{ij}\right)$ Using the property in Equation (9), it can be shown that the expectation and the second moment of w_i satisfy: $\mathbb{E}[w_i] = x^T y$ and

$$\mathbb{E}[w_i^2] = 2(x^T y)^2 + ||x||_2^2 ||y||_2^2 + \Sigma_{j=1}^N \frac{1}{g_j} x_j^2 y_j^2 - 3\Sigma_{j=1}^N x_j^2 y_j^2.$$

Since $u^T v = \frac{1}{\ell}\sum_{i=1}^{\ell} w_i$, using the above result, we can show that:

$$Var(u^T v) = \frac{1}{\ell}((x^T y)^2 + ||x||_2^2 ||y||_2^2 + \Sigma_{j=1}^N \frac{1}{g_j} x_j^2 y_j^2 - 3\Sigma_{j=1}^N x_j^2 y_j^2).$$

In order to prove proposition 2, we need the following lemma.

Lemma 1. *Consider a data vector $x \in \mathbb{R}^N$ which satisfies condition (5). Let $y \in \mathbb{R}^{N \times N}$. Consider a sparse random matrix $\Phi \in \mathbb{R}^{\ell \times N}$ satisfies condition (9), with sparsity parameter $\rho = g_j$. Define $\ell = O(\frac{1+\gamma}{\epsilon^2}\mu^2 \log N \Sigma_{j=1}^N \frac{1}{g_j})$. The random projections $\frac{1}{\sqrt{\ell}}\Phi u$ and $\frac{1}{\sqrt{\ell}}\Phi v_i$ then produces an estimation $\hat{a}_i$ for $x^T y_i$, with probability at least $1 - N^{-\gamma}$, satisfying $|\hat{a}_i - x^T y_i| \leq \epsilon ||x||_2 ||y_i||_2$, $\forall_{1 \leq iN}$.*

Proof (Proof of Lemma 1). Due to lack of space, the proof for Lemma 71 cannot be included. The proof is similar to that of Theorem 1 in [20] except that the term $(s-3)$ in [20] is replaced by $(\sum_{j=1}^N \frac{1}{g_j} - 3)$.

Proof (Proof of Proposition 2). Consider an orthonormal transform $\Psi \in \mathbb{R}^{N \times N}$. Let the transform coefficients $\theta = [x^T \psi_1, ..., x^T \psi_N]^T$. Let us order the transform coefficients θ in decreasing of magnitude, i.e., $|\theta|_{(1)} \geq |\theta|_{(2)} .... \geq |\theta|_{(N)}$.

The approximation error by taking the largest k coefficients in magnitude, and setting the remaining coefficients to zero can therefore be given by $||\theta - \hat{\theta}_{opt}||_2^2 = \Sigma_{i=k+1}^{N} |\theta|_{(i)}^2$. Let $||\theta - \theta_{opt}||_2^2 \leq \eta ||\theta||_2^2$ and assume that x satisfies condition (5), with positive integer, $\ell = O(\frac{1+\gamma}{\beta^2}\mu^2 \log N \Sigma_{j=1}^{N} \frac{1}{g_j})$. The random projections $\frac{1}{\sqrt{\ell}}\Phi u$ and $\{\frac{1}{\sqrt{\ell}}\Phi\psi_1, ..., .\frac{1}{\sqrt{\ell}}\Phi\psi_n\}$ thus could produce estimates $\{\hat{\theta_1}, ..., \hat{\theta_N}\}$, where the estimates satisfy $|\hat{\theta}_i - \theta_i| \leq \beta ||\theta||_2$ with high probability (Lemma 1).

Now ordering the estimates $\hat{\theta}$ in decreasing magnitude, we define our approximation $\tilde{\theta}$ as keeping the k largest (in magnitude) components of $\hat{\theta}$, and setting the other components to zero. It can be shown that [20] for $\beta = O(\frac{\epsilon\eta}{k})$, the approximate error is $||x - \hat{x}||_2^2 = (1+\epsilon)\eta ||x||_2^2$. Therefore the number of random projections we need can be given by

$$\ell = O(\frac{1+\gamma}{\epsilon^2 \eta^2} k^2 \mu^2 \log N \Sigma_{j=1}^{N} \frac{1}{g_j}).$$

An Adaptive Strategy for Energy-Efficient Data Collection in Sparse Wireless Sensor Networks

Mario Di Francesco[1], Kunal Shah[2], Mohan Kumar[2], and Giuseppe Anastasi[3]

[1] Center for Research in Wireless Mobility and Networking (CReWMaN)
University of Texas at Arlington
mariodf@uta.edu
[2] Pervasive and Invisible Computing (PICO) Lab
University of Texas at Arlington
kshah@cse.uta.edu, mkumar@uta.edu
[3] Pervasive Computing and Networking Laboratory (PerLab)
University of Pisa, Italy
giuseppe.anastasi@iet.unipi.it

Abstract. Sparse wireless sensor networks (WSNs) are being effectively used in several applications, which include transportation, urban safety, environment monitoring, and many others. Sensor nodes typically transfer acquired data to other nodes and base stations. Such data transfer operations are critical, especially in sparse WSNs with mobile elements. In this paper, we investigate data collection in sparse WSNs by means of special nodes called Mobile Data Collectors (MDCs), which visit sensor nodes opportunistically to gather data. As contact times and other information are not known a priori, the discovery of an incoming MDC by the static sensor node becomes a critical task. Ideally, the discovery strategy should be able to correctly detect contacts while keeping a low energy consumption. In this paper, we propose an adaptive discovery strategy that exploits distributed independent reinforcement learning to meet these two necessary requirements. We carry out an extensive simulation analysis to demonstrate the energy efficiency and effectiveness of the proposed strategy. The obtained results show that our solution provides superior performance in terms of both discovery efficiency and energy conservation.

1 Introduction

Wireless sensor networks (WSNs) have become an enabling technology for a wide range of applications [1]. WSNs are based on sensor nodes which are constrained in terms of their resources: energy, computational power and radio bandwidth. They normally operate in uncertain and dynamic environments where the state of the system changes considerably over time. For example, in data collection applications, uncertainty exists due to intermittent links or traffic conditions. Moreover, the network itself is dynamic due to such events as node mobility and depleted battery. WSN applications need to cope with such dynamic and

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 322–337, 2010.

uncertain conditions inherent in sensor networks, while simultaneously achieving application-specific QoS requirements. As a consequence, adaptive resource management is a key to any successful middleware solution enabling such applications [2].

In the context of environmental monitoring, a large number of sensor nodes are typically deployed over a geographical area to form a dense ad hoc network. Sensors use multi-hop communication to send data acquired from the external environment to a sink node or to an Access Point (AP) in the infrastructure. However, several environmental monitoring applications do not require fine-grained sensing. Examples of such applications include monitoring of weather conditions in large areas, air quality in urban scenarios, terrain conditions for agriculture, and so on. In this case, it is possible to consider a *sparse wireless sensor network*, i.e., a WSN where the density of nodes is so low that they cannot communicate each other through multi-hop paths, or even directly. In order to make communication feasible, data collection in sparse WSNs can be accomplished by means of *mobile data collectors* (MDCs). MDCs are special mobile nodes responsible for data gathering and/or dissemination. They are assumed to be powerful in terms of data storage and processing capabilities, and are not energy constrained, in the sense that their energy source can be replaced or recharged easily. An MDC can serve either as a Mobile Sink (MS), a mobile node which is also the endpoint of data collection, or as a Mobile Relay (MR), which carries data from sensors to the sink node or an infra-structured AP. In either role, the MDC moves throughout the WSN, and in most cases it is autonomous.

Sparse WSNs with MDCs have many advantages if compared to traditional dense WSNs. First, costs are reduced, since fewer nodes can be deployed, as there is no need for a connected network. Second, as data is collected directly by the MDC from sensor nodes, reliability is improved as a result of less congestion and collisions. Finally, data collection by MDC can extend the WSN lifetime, as the energy consumption is spread more uniformly in the network with respect to dense (static) WSN, where the nodes close to the sink are usually more loaded than the others. However, the data collection paradigm in sparse WSNs with MDCs is different, and introduces significant challenges. Among them, we can mention contact detection and mobility-aware energy conservation schemes.

Communication between an MDC and sensor nodes takes place in two phases. First, sensor nodes discover the presence of the MDC in their communication range. Then, they can transfer collected data to the MDC while satisfying certain reliability constraints, if required. Unlike MDCs, sensor nodes have a limited energy budget, so that the data-collection process has to be energy efficient in order to prolong network lifetime. In addition, such energy-conserving mechanisms should not compromise the timeliness of communication. This is critical especially when the MDC has only a short contact time with sensors, and also in the case when such contacts cannot be predicted accurately. In fact, a major problem in data collection is that sensor nodes usually do not have *a priori* knowledge of the MDC mobility pattern. Furthermore, even in cases where the arrivals can be predicted, there is a chance that the MDC contacts can be

affected by delays or can change their rate. Hence robust and flexible mechanisms have to be defined in order to adapt to operating conditions autonomously. To this end, mechanisms based on artificial intelligence are very appealing, since they are flexible and robust.

In this paper, we address the problem of the MDC discovery by exploiting reinforcement learning, a branch of artificial intelligence targeted to unsupervised learning. We define discovery and data transfer protocols for energy-efficient data collection in sparse WSNs with MDCs, and propose an adaptive strategy exploiting a middleware framework based on *Distributed Independent Reinforcement Learning* (DIRL). The proposed solution is specifically targeted to energy-aware resource allocation in sparse WSNs with MDCs. The remainder of the paper is organized as follows. Section 2 presents an overview of the related work, while Section 3 introduces the system model and the reference scenario. Section 4 describes the data collection application built on top of the DIRL framework. Section 5 outlines the simulation setup and introduces relevant metrics for evaluation. Section 6 discusses the obtained results. Finally, Section 7 concludes the paper.

2 Related Work

Solutions for energy-efficient data collection and adaptive resource management in sparse WSNs have already been proposed in the literature. However, in most cases these two issues have been considered separately. Accordingly, we will consider the relevant literature in different sections below.

Several researchers have investigated data collection in sparse WSNs. In [3,4,5] data collection is performed by autonomous MDCs. A data collection scheme is presented in [3], under the assumption that the MDC has a completely predictable mobility. The problem of data collection has been considered also in [4], where both discovery and data transfer are characterized, and the mobility pattern of the MDC is assumed to follow a Poisson distribution. An extensive characterization of data collection in sparse WSNs is provided in [5], where the impact of discovery on data transfer is evaluated. The major limitation of these solutions is that they assume static operating parameters. An adaptive data collection strategy has been proposed in [6], but the approach does not consider the problem of discovery. Solutions based on reinforcement learning have been proposed in [7,8] for the context of WSNs with mobile elements. However the focus of [7] is more on routing rather than discovery, since the WSN is assumed to be rather dense. On the other side, [8] actually exploits reinforcement learning for discovery, but in the different context of sparse WSNs where nodes operate as peers.

As for middleware solutions, while many papers such as [9,10,11] have addressed resource management in dense WSN, there are only a few works specifically targeted to the same problem in the different context of sparse WSNs. Among them, the Impala [12] middleware architecture has been proposed for application adaptation and update. However, Impala has been specifically targeted to scenarios where all nodes are mobile and act as peers. In addition, the

focus is more on application reconfiguration rather than on resource allocation. More recently, the TinyLime [13] middleware has been proposed for the specific scenario of sparse WSNs. TinyLime – which is based on a tuple space model – provides mechanisms to perform data aggregation and tune the activity of nodes in order to save energy. However, the focus of TinyLime is on the proposed programming abstraction rather than on adaptation and resource management. On the contrary, in this paper we propose an adaptive middleware approach to resource allocation for energy-efficient data collection in sparse WSNs.

3 System Overview

Before introducing the adaptive data collection strategy, it is necessary to present the elements on top of which our proposed strategy is built. First we present two background elements necessary to describe our proposed strategy: (i) a reference scenario; and (ii) the DIRL middleware framework that will be employed for data collection applications.

3.1 Network Scenario

The reference network scenario is depicted in Figure 1(a). Specifically, we will consider a single MDC and assume that the network is sparse so that, at any time, the MDC can communicate with at most one static node.

Data collection takes place only during a *contact*, i.e., when the static node and the MDC can reach each other. Furthermore, the area within the communication range of the static node is called *contact area*, and the overall time spent by the MDC inside the contact area is called *contact time*. During a contact, messages exchanged between the MDC and the static node experience a certain message loss, denoted by $p(t)$. We also assume that the MDC mobility is not controllable, and we define as *tour* (and denote it with T) the smallest time duration after which the mobility pattern repeats [7]. On the other side, we define as *inter-contact time* the actual period of time elapsed from the beginning of a contact to the beginning of the subsequent one.

The overall data collection process can be split into three main phases. Figure 1(b) shows the state diagram of the static sensor node [14]. As MDC arrivals are generally unpredictable, the static node performs a discovery phase for the timely detection of the MDC. Upon detecting the MDC, the static node switches from the discovery state to the data transfer state, and starts transmitting data to the MDC. At the end of the data transfer phase, the static node may switch to the discovery state again in order to detect the next MDC passage. However, if the MDC has a (even partially) predictable mobility, the static node can exploit this knowledge to further reduce its energy consumption [14]. In this case, the static node can go to sleep until the next expected arrival of the MDC.

Similar to [5], we will use an asynchronous discovery protocol and an ARQ-based protocol for data transfer. In detail, the MDC periodically sends special messages called *beacons* to advertise its presence in the surrounding area. The

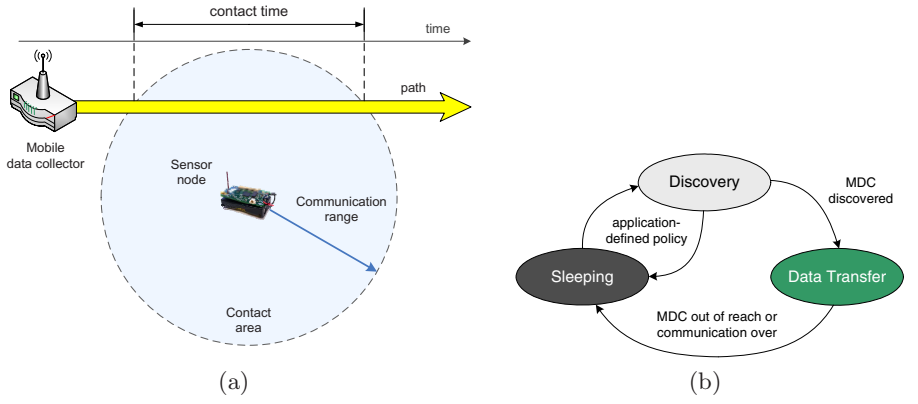


Fig. 1. Reference scenario (a) and state diagram for the static node (b)

duration of a beacon message is equal to T_{BD}, and subsequent beacons are spaced by a *beacon period*, indicated with T_B. In order to save energy during the discovery phase, the static node operates with a duty-cycle δ, whose active time $T_{ON} \geq T_B + T_{BD}$ so that a complete beacon can be received during the active time, provided that it wakes up when the MDC is in the contact area.

As soon as it receives a beacon from the MDC, the static node enters the data transfer state. While in this state, the static node remains always active to exploit the contact as much as possible. On the other hand, the MDC enters the data transfer phase as soon as it receives the first message sent by the static node, and stops beacon transmissions. The communication protocol adopted during the data transfer phase is *selective repeat* [15], i.e., a window-based ARQ protocol with selective retransmission, whose window size is assumed to be equal to W messages. Note that the acknowledgement messages in the ARQ scheme are used not only for implementing a retransmission strategy, but also as an indication of the MDC presence in the contact area.

The data transfer phase ends either when the static sensor has no more messages to transmit during a contact, or the MDC is not reachable any more. However, since the static node generally does not know when the MDC will leave the contact area, it assumes that the MDC has exited the contact area when it misses N_{ack} consecutive acknowledgments. Similarly, the MDC assumes that the communication is over when it does not receive any message in a given period of time.

3.2 Distributed Independent Reinforcement Learning (DIRL)

Distributed Independent Reinforcement Learning (DIRL) [16] is a framework that exploits Q-learning [17] to enable autonomous and adaptive applications with inherent support for efficient resource management. The main idea of DIRL is to allow each individual sensor node to self-schedule its tasks and allocate its

resources by learning their usefulness (i.e., their utility) in any given state while honouring application defined constraints and maximizing total amount of reward over time. Q-learning demands minimal computational resources and does not require a model of the environment in order to operate. Hence it is ideal for implementations running on resource-constrained sensor nodes. In addition, DIRL is based on independent learning where each agent applies the learning algorithm in a classic sense (like a single agent system) and ignores the presence of other agents. As a consequence, each sensor node can autonomously and dynamically self-configure in order to maximize its own reward. The main advantage of using independent learning in DIRL is that no coordination is required among sensor nodes, which is beneficial to scenarios where sensors are sparsely deployed.

DIRL uses a simple single step immediate reward and uses a simple weighted hamming distance between two states in order to reduce the state space, which otherwise would be unaffordable for constrained sensor nodes. In addition, DIRL uses the classic exploration and exploitation strategy used in most approaches based on reinforcement learning to get the utilities of the individual tasks. Instead of using the original DIRL exploration policy, we consider a mobility-aware exploration probability based on the number of contacts. More specifically, it is given by

$$\epsilon = \epsilon_{min} + \max\left(0, k \cdot (c_{max} - c)/c_{max}\right)$$

where ϵ_{max} and ϵ_{min} define upper and lower boundaries for the exploration factor, respectively; c_{max} represents the maximum number of contacts (as obtained from the application) after which a steady state condition is likely to be reached, while c represents current number of detected contacts; finally, k is a constant that can be tuned to control the descending rate to the minimum exploration probability. Therefore, the heuristic presented above allows initial exploration with a higher rate and gradually decreases over time as DIRL is able to detect up to c_{max} contacts. Note that some minimum exploration is always required, so as to allow a sensor node to dynamically reconfigure in case of environmental changes.

DIRL needs the following as inputs from the application:

- A set of tasks to be executed, in some priority order. Note here that priority is important only until Q-values, i.e., the learned utilities for all actions in each state, are not established or if two tasks have similar Q-values.
- An applicability predicate associated with each task, incorporating both application-specific constraints and reward functions.
- A state representation consisting of both system and application variables, along with the corresponding weights for deriving the distance between states, and aggregating similar ones.
- The maximum number of states that DIRL should try to explore. This gives an upper bound on number of states in the system. In case of need, a state replacement policy might be introduced, or the hamming distance threshold might be tweaked to accommodate new states into existing (similar) ones.

After obtaining the input from the application, DIRL executes the following algorithm (depicted in Figure 2). Initially all Q-values are set to zero. At each

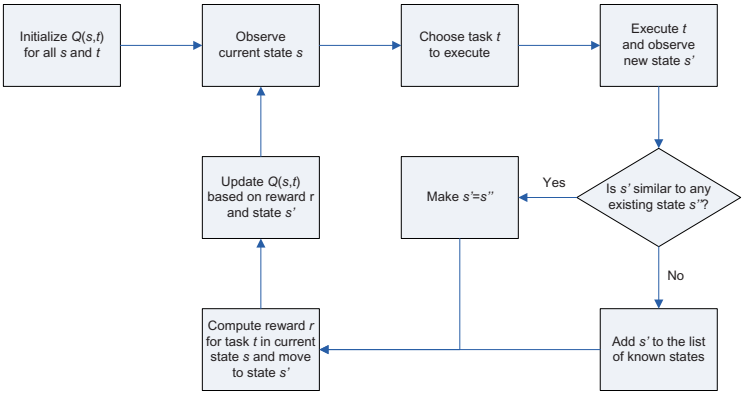


Fig. 2. State diagram for the static node during data collection

time-step DIRL selects a task to execute based on the exploration/exploitation strategy as described earlier. Exploration selects an available task randomly, while exploitation selects the best task according to the learned utilities, i.e., the Q-values. After the execution of a task, DIRL observes the new state s' and compares it with all existing states based on a hamming distance. Finally, DIRL computes the reward for the executed task in state s' and updates the corresponding Q-values.

4 Adaptive Data Collection (ADC) Strategy

In this section we define an adaptive strategy based on DIRL for energy-efficient (and reliable) data collection in sparse WSNs. The goal of this strategy is to maximize the percentage of data successfully transferred during contacts, while minimizing the energy consumption of sensor nodes, even in scenarios where the mobility pattern of the MDC is not known in advance. To this end, we defined the different the tasks to be used by DIRL with reference to the discovery phase only.

In the context of the reference scenario already introduced in Section 3.1, we have identified three major tasks, each corresponding to a different duty-cyle used for discovering the MDC. In order to make the derivation of tasks more general, we have defined the actual duty-cycles on the basis of a maximum allowed duty-cycle, denoted as δ_{max}.

- *High Duty-Cycle* (HDC). The static sensor is executing a HDC, equal to δ_{max}. Ideally this task should be executed whenever the probability of MDC being in the contact area is high.
- *Low Duty-Cycle* (LDC). The static sensor is executing a LDC, equal to $0.5 \cdot \delta_{max}$. Ideally this task should be executed whenever the probability of MDC being in the contact area is low, so that the correspondent energy consumption is very low as well.

– *Very Low Duty-Cyle* (VLDC). The static node executes a VLDC, equal to $0.1 \cdot \delta_{max}$. Ideally this task should be executed whenever the probability of MDC being in the contact area is very low, so that the correspondent energy consumption can be considered as almost negligible with respect to the maximum allowed duty-cycle.

As can be seen from the above-mentioned task definitions, the MDC discovery and the successful data transfer process can be maximized while minimizing the energy usage if we can adaptively schedule above tasks based on learned probability of MDC being in contact. DIRL learns this probability in the form of utilities built by using local rewards. In order to implement the adaptive data collection strategy, DIRL executes the discovery tasks according to the algorithm depicted in Figure 2 and already presented in Section 3.2.

The data transfer is executed as a different process, in the sense that it is not a DIRL task. In order to manage this, we have introduced a state variable i_c which is true when the MDC is assumed to be in contact with the static sensor. In detail, i_c is set to one when the discovery phase ends with success, i.e., a beacon is successfully received by the static sensor. Moreover, i_c is set to zero when the static sensor has lost a number N_{ack} of consecutive acknowledgement messages as a result of the data transfer phase, thus assuming that the MDC has exited the contact area. Hence, the data transfer phase can be entered only after the MDC has been detected (i.e., $i_c = 1$), under the constraint that messages in transmission[1] are enough to fill a complete window. On the other hand, discovery tasks can always be executed.

For all tasks scheduled by static node, the reward is defined as $r_t = i_c \cdot e_p - e_s$, where i_c is the contact state variable, e_p is the expected price, and e_s the energy spent. Note that the expected price is chosen as a multiple of the energy spent for that task, so as to allow a symmetric evaluation of the reward function. Thus, for each task, the reward is equal to the expected price e_p minus the energy spent e_s if the MDC has been successfully detected, otherwise it is equal to minus e_s.

In order to map the presence of the MDC to the specific instants where it is in contact, we have introduced a temporal characterization in the state representation of the static sensor nodes. On the basis of the concept of tour, we split the time (as perceived by a static sensor) into a number of intervals called *time domains*, whose duration is denoted as T_d. More specifically, each task is scheduled for one time domain, at the end of which utilities are updated and the sensor node evaluates the new state. The granularity of time domain length T_d represents a trade-off between the storage and computational requirements at the static sensor, and the efficiency of DIRL. The lower the value of T_d, the higher is the accuracy of DIRL to schedule the duty-cycle tasks with a fine granularity. On the other hand, a higher number of states increases the overall computation requirements of the learning algorithm.

As all statistics regarding the mobility pattern of the MDC are estimated by the static sensors, we added some filtering techniques to avoid misinterpretation

[1] Messages in transmission can be either buffered messages or messages which have been already transmitted but not yet acknowledged.

of context. For instance, the static sensor might consider a single actual MDC contact as multiple observed contacts. To this end, we implemented a simple timeout technique, so that the static sensor considers the successful reception of a beacon message as a new contact only when a certain time has elapsed since the preceding contact detection. We set this timeout value to 30 s. Similarly, it is required to handle missed contacts, i.e., an actual contact not being detected by the static sensor, as this would result in incorrect learning of the MDC mobility pattern. For this reason, we maintained a short history of contacts (in terms of the time domains where they occur), and adjusted the state evaluation accordingly.

5 Simulation Setup

In this section we will evaluate the performance of the DIRL-based adaptive strategy introduced in the previous section. To this end, we will consider the following performance metrics.

- *Activity ratio*, defined as the ratio between the active time and the total time spent during discovery[2].
- *Discovery ratio*, defined as the average of the ratio between the number of contacts correctly detected by the static sensor and the total number of contacts.
- *Energy efficiency*, defined as the mean energy spent by the static sensor per each message (or byte) correctly transferred to the MDC.

As for the energy expenditure, we implemented a simple model that characterizes the radio, while we do not address the energy expenditure of the CPU, since it is almost negligible. Specifically, the energy expenditure of the radio is calculated as $P_{state} \cdot T_{state}$, where P_{state} and T_{state} denote respectively, the power consumption of the radio and the amount of time spent in a given state, i.e., receive, transmit and sleep. We assume that the energy consumption of the radio during idle periods, i.e., when it is monitoring the channel, is the same as in the receive state. As for message loss, we used the model considered in [6,5] and based on experimental data measured in a real testbed in the same scenario [18].

In order to compare the performance of DIRL with other approaches, we also considered the following schemes.

- *Random*. At each time step a static node executes a task chosen at random between available ones.
- *SORA*. Static nodes use the heuristic-based reinforcement learning defined in [11].

[2] In this metric we do not consider the activity due to data transfer. Hence, the activity ratio is a measure of the average duty-cycle which derives from the executions of the different discovery tasks.

Table 1. Parameters used for simulation

Parameter	Value	Parameter	Value
Minimum exploration (ϵ_{min})	0.1	Beacon duration (T_{BD})	10 ms
Maximum exploration (ϵ_{max})	0.3	Window size (W)	16
Descending rate (k)	0.2	Consecutive lost acks (N_{ack})	5
Maximum contacts (c_{max})	10	Message payload size	24 bytes
Time domain duration (T_d)	100 s	Frame size	36 bytes
Message generation interval	10 s	Radio transmit power (0 dBm)	49.5 mW
Expected price (e_p) multiplier	10	Radio receive/idle power	28.8 mW
Beacon period (T_B)	100 ms	Radio sleep power	0.6 μW

– *Oracle*. Static nodes have perfect knowledge on MDC contacts, so they do not perform discovery at all. They start transmitting data as soon as the MDC is in the contact area and stop transmitting when there are no more data or the MDC is out of contact.

As for the mobility pattern of the MDC, we considered three different scenarios.

– *Deterministic mobility*. The MDC arrivals are periodic, the inter-contact time is fixed. This mobility pattern corresponds to the case where the arrivals of the MDC are known in advance, e.g. when the MDC is a shuttle [3].
– *Gaussian mobility*. The MDC arrivals are periodic, the inter-contact time follows a normal distribution with given mean and variance. This mobility pattern corresponds to the case where the MDC arrivals are rather predictable, but suffer from a certain spread [14]. This can be the case of cars which are affected by traffic conditions.
– *Poisson mobility*. The MDC arrivals are periodic, the inter-contact time is exponential. This mobility pattern corresponds to the case where MDC arrivals are rather unpredictable [4].

We carried out a performance evaluation by using a discrete event simulator written in Java. To derive confidence intervals we used the replication method with a 95% confidence level. In all experiments we performed 10 replicas, each consisting of at least 1000 MDC passages. In the following, we will assume a MICA2 series mote [19] as the static node, and use the related parameters for power consumption. We will assume that the radio is operating at a link speed of 19.6 kbps bitrate. All other simulation parameters, chosen according to the methodology used in [5], are summarized in Table 1.

6 Simulation Results

In order to evaluate the performance of the DIRL-based ADC strategy, we split simulations in two parts. In the first one, we investigated how the proposed approach reacts to dynamic (transient) conditions., while in the second one we focused on the performance in steady-state conditions. For the sake of clarity, in the following we will consider a single MDC which collects data from a single static sensor node.

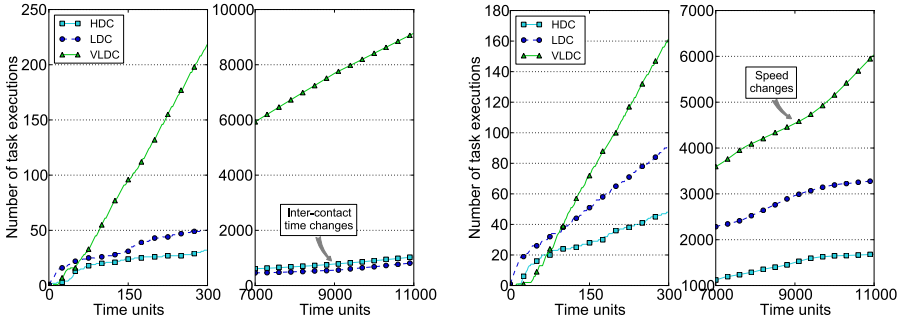


Fig. 3. Number of task executions over time for variations in inter-contact time (a) and speed (b) of the MDC

6.1 Analysis in Dynamic Conditions

As for the analysis in dynamic conditions, we considered two different kinds of variation.

- *Variation in the inter-contact time.* The MDC moves at 20 km/h and starts visiting the sensor node every 1800 s. Then, after some time, the inter-contact time changes to 900 s. This scenario has been considered as representative when the MDC increases the rate of visits, for instance, because there is a need to get fresh data more frequently from the environment.
- *Variation in the speed of the MDC.* The MDC completes a tour in 1800 s and starts visiting the sensor node at a speed of 3.6 km/h. Then, after some time, it changes its speed to 40 km/h while keeping constant the inter-contact time. This scenario has been considered to stress the ability of system to discover the MDC when the contact duration suddenly decreases significantly.

In both cases we set the duration of the simulation to 18000 time units[3] (corresponding to 500 hours of simulated time), and assumed that the variation takes place after 9000 time units (corresponding to 250 hours from the beginning of the experiment). We analyze the ability of our approach to adapt to the operating conditions by means of the (relative) number of task executions as a function of time. We show one representative simulation run for each kind of variation in Figures 3(a) and 3(b) (we have verified that the trend is almost the same also when additional replicas are performed). To highlight the initial learning phase and the reaction to the variation we split the horizontal axis into two parts.

We start considering the variation in the inter-contact time, as shown in Figure 3(a). During the initial phase, where exploration is performed more than

[3] For convenience, we denoted as a *time unit* the interval corresponding to the duration of a time domain, which is equal to 100 s in our experiments.

exploitation, the LDC task is executed more than the other two. However, in steady state conditions (e.g., after 7000 time units), the VLDC task gets the highest number of executions, followed by the HDC and the LDC tasks, respectively. The VLDC task has a high slope, different from the other two tasks. In addition, the number of executions in a HDC task is always higher than those in a LDC task. This happens because the contact time is relatively short, so that it is more convenient (in terms of rewards) to execute the HDC task during the time domains where the MDC is actually in contact with the static node.

In Figure 3(a), it can be noticed that the ADC strategy actually adapts to the new parameters after 9000 time units, when the inter-contact time reduces. More specifically, the VLDC task executions are decreased while the other two tasks are performed more often. This is related to the fact that the inter-contact time is shorter, so that the distance (in terms of time domains) between executions of task deriving from exploitation is shorter. This changes the relative frequency of executions of all tasks.

On the other side, the variation in the MDC speed is shown in Figure 3(b). During the initial phase, the LDC task gets the highest number of executions. After a while, similar to the previous case, the VLDC task increases its executions significantly. In stationary conditions, the LDC task is executed more often than the HDC task. In addition, the three plots are closer with respect to the previous case. This is because, when the contact time is longer (when the speed is 3.6 km/h), actually there is little difference between the different tasks. In fact, almost all contacts are detected irrespective of the duty-cycle used for discovery.

The adaptation is also apparent when the MDC speed changes after 9000 time units. In contrast to the previous case, the executions of the VLDC task are increased, while the (higher) duty-cycle tasks are performed less frequently.

6.2 Analysis in Stationary Conditions

In this section we evaluate the performance of the DIRL-based ADC strategy in stationary conditions. This gives an indication of how the proposed solution is able to perform when the network is stationary, for instance when it has reached its steady state conditions. Whenever not specified otherwise, we assume that the mobility pattern is deterministic with a 1800 s inter-contact time. All other parameters are as specified in Table 1.

In the first set of experiments we compared our approach – in terms of activity ratio, discovery ratio and energy efficiency – to the other middleware schemes already presented in Section 5 for different speeds of the MDC. As for the activity ratio, from Figure 4(a) we can see that DIRL performs much better than Random and SORA for all MDC speeds. Actually, we can see that the highest activity ratio is obtained for the lowest MDC speed in all cases. That is because when the speed of the MDC is 3.6 km/h, the contact time is long enough so that the discovery tasks gets rewarded in a larger number of time domains, resulting in a higher discovery task being executed more times. When contacts get detected in a lower number of time domains, which happens at the higher speeds, the

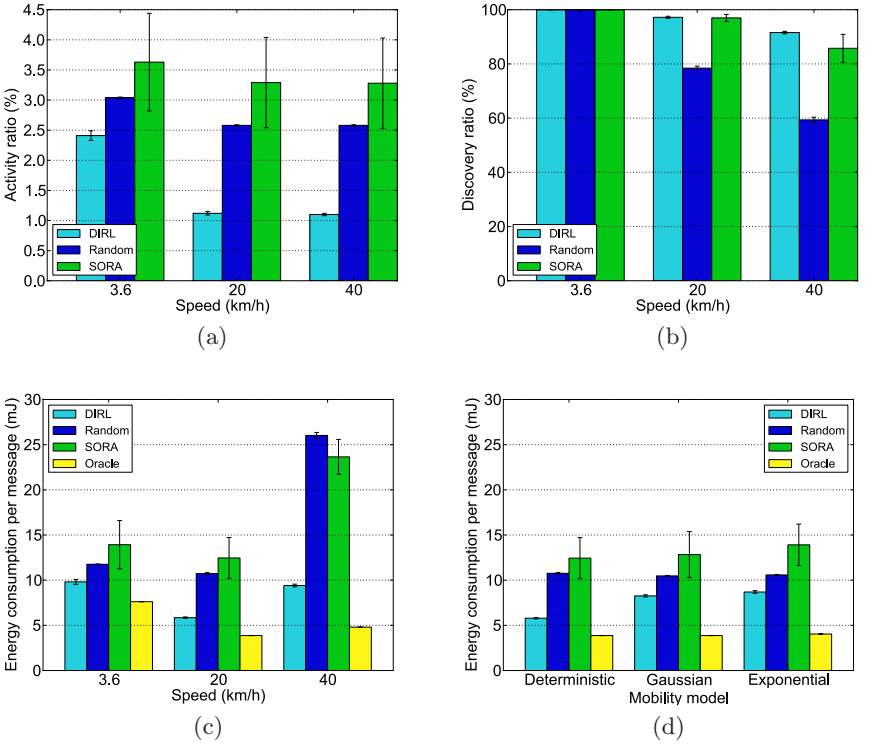


Fig. 4. Activity ratio (a), discovery ratio (b) and energy consumption (c) as a function of the MDC speed for different middleware schemes. Energy consumption as a function on the mobility pattern for different middleware schemes (d).

activity ratio is actually lower. Apart from this, the results show that DIRL can get a very low activity ratio, in the order of a few percent, and outperforms other approaches such as Random and SORA. More specifically, it seems that SORA cannot efficiently exploit the limited number of rewards in the considered scenario.

The activity ratio alone is not a measure of discovery efficiency, since contacts may be missed as a result of sensors being asleep for most of the time. To this end we considered the discovery ratio, which is given in Figure 4(b). We can see that when the mobility is low, almost all contacts are detected, independent from the adopted middleware scheme. The situation is different, however, when the speed is high (i.e., 20 or 40 km/h). In this case the two middleware schemes based on reinforcement learning (i.e., both DIRL and SORA) clearly get better results than the Random approach. There also is a slight improvement of DIRL over SORA, since it obtains a discovery ratio always over 90%.

The most important metric, indeed is energy efficiency, which can characterize the joint effect of discovery and data transfer. In fact, the discovery efficiency does not give an indication of how effective the discovery is for the data transfer

(in other terms, if contacts are efficiently exploited to transfer buffered data). The results are provided in Figure 4(c) where the energy consumption per correctly transferred message is shown (we also show here the Oracle scheme as a reference). We can see that, as a general trend, the energy expenditure is higher for speeds of 3.6 km/h and 40 km/h, compared to the intermediate speed of 20 km/h. This is against the expected increase in energy expenditure with MDC speed due to the reduction in contact times and also the probability of successful discover. In the 3.6 km/h scenario under the evaluated conditions, the message generation rate is very low with respect to the contact time. As a consequence, since all middleware approaches are performing discovery tasks all the time, they end up completing transfers prematurely, by performing unnecessary discovery. Besides this consideration, the figure clearly shows the advantages of DIRL over the other approaches. DIRL performs much better than SORA because it can exploit the contact more efficiently, in terms of the time actually available for data transfer (and also as throughput per detected contact).

In the second set of experiments, we fixed the speed to the intermediate value of 20 km/h and evaluated the performance of the middleware schemes on the basis of different mobility patterns. For the sake of space, we will focus only on the energy expenditure per transmitted message, which is depicted in Figure 4(d). The goal of this set of experiments is to evaluate how the uncertainty related to the MDC mobility affects the energy consumption. To this end, we ordered the mobility patterns in increasing level of uncertainty: deterministic, Gaussian with a 30 s spread over the mean, and exponential. All mobility patterns use a 1800 s (average) inter-contact time.

As expected, the activity ratio increases when the uncertainty on the mobility pattern of the MDC increases for all middleware schemes. In all cases DIRL performs better than other approaches. This is because DIRL can tune discovery to the actual demand better than the other schemes. In any case, the variance in the energy expenditure is lower for DIRL rather than for SORA, as the former tracks contacts more accurately and efficiently.

In conclusion, proposed solutions using MDC can be effectively used in a wide range of scenarios, even when the contact time is short and the uncertainty on MDC arrivals is high. Thus DIRL results in effective resource allocation while, at the same time, exhibiting very good performance.

7 Conclusions

In this paper, a novel Adaptive Data Collection (ADC) strategy for sparse Wireless Sensor Networks (WSNs) with Mobile Data Collectors (MDCs) is proposed. The problem of energy-efficient MDC discovery by exploiting the Distributed Independent Reinforcement Learning (DIRL) framework has been addressed. Our results show that the ADC strategy results in efficient resource allocation, in terms of both low activity needed for discovery and a high data transfer efficiency. Compared to existing solutions, the proposed approach not only performs better, but also can adapt to different operating conditions and mobility patterns

characterized by high uncertainty. As a result, it can be effectively used in the context of sparse WSNs where rewards may be very limited.

Our work can be extended along different directions: (i) define a better characterization of the MDC mobility pattern so that time-domains are automatically derived or tuned; (ii) incorporate information resulting from the data transfer phase into the reward to allow exploitation of the feedback from data collection phase to improve discovery. In addition, we will implement the ADC strategy on real sensor hardware and perform experiments in a WSN testbed.

Acknowledgments

This work has been carried out while Mario Di Francesco was with the Department of Information Engineering, University of Pisa, Italy.

The research was funded partially by the US National Science Foundation awards CSR 0834493 and CNS 0721951, and partially by the Italian Ministry for Education and Scientific Research (MIUR) under the FIRB ArtDeco and PRIN WiSe DeMon projects.

References

1. Anastasi, G., Conti, M., Di Francesco, M., Passarella, A.: Energy conservation in wireless sensor networks: A survey. Ad Hoc Networks 7(3), 537–568 (May 2009)
2. Hadim, S., Mohamed, N.: Middleware challenges and approaches for wireless sensor networks. IEEE Distributed Systems Online 7(3) (March 2006)
3. Chakrabarti, A., Sabharwal, A., Aazhang, B.: Using predictable observer mobility for power efficient design of sensor networks. In: Zhao, F., Guibas, L.J. (eds.) IPSN 2003. LNCS, vol. 2634, pp. 129–145. Springer, Heidelberg (2003)
4. Jain, S., Shah, R., Brunette, W., Borriello, G., Roy, S.: Exploiting mobility for energy efficient data collection in wireless sensor networks. ACM/Springer Mobile Networks and Applications 11(3), 327–339 (2006)
5. Anastasi, G., Conti, M., Di Francesco, M.: Reliable and energy-efficient data collection in sparse sensor networks with mobile elements. Performance Evaluation 66(12), 791–810 (December 2009)
6. Anastasi, G., Conti, M., Monaldi, E., Passarella, A.: An adaptive data-transfer protocol for sensor networks with Data Mules. In: WoWMoM 2007: Proceedings of the 8th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, pp. 1–8 (2007)
7. Baruah, P., Urgaonkar, R., Krishnamachari, B.: Learning-enforced time domain routing to mobile sinks in wireless sensor fields. In: LCN 2004: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, pp. 525–532 (2004)
8. Dyo, V., Mascolo, C.: Efficient node discovery in mobile wireless sensor networks. In: Nikoletseas, S.E., Chlebus, B.S., Johnson, D.B., Krishnamachari, B. (eds.) DCOSS 2008. LNCS, vol. 5067, pp. 478–485. Springer, Heidelberg (2008)
9. Heinzelman, W., Murphy, A., Carvalho, H., Perillo, M.: Middleware to support sensor network applications. IEEE Network 18(1), 6–14 (January 2004)

10. Marron, P.J., Lachenmann, A., Minder, D., Hahner, J., Sauter, R., Rothermel, K.: TinyCubus: a flexible and adaptive framework sensor networks. In: EWSN 2005: Proceedings of the 2nd European Workshop on Wireless Sensor Networks, pp. 278–289 (2005)
11. Mainland, G., Parkes, D.C., Welsh, M.: Decentralized, adaptive resource allocation for sensor networks. In: NSDI 2005: Proceedings of the 2nd Symposium on Networked Systems Design & Implementation, pp. 315–328 (2005)
12. Liu, T., Martonosi, M.: Impala: a middleware system for managing autonomic, parallel sensor systems. In: PPoPP 2003: Proceedings of the 9th ACM SIGPLAN symposium on Principles and practice of parallel programming, pp. 107–118 (2003)
13. Curino, C., Giani, M., Giorgetta, M., Giusti, A., Murphy, A.L., Picco, G.P.: Mobile data collection in sensor networks: The TinyLime Middleware. Elsevier Pervasive and Mobile Computing Journal 4(1), 446–469 (December 2005)
14. Jun, H., Ammar, M., Zegura, E.: Power management in delay tolerant networks: A framework and knowledge-based mechanisms. In: SECON 2005: Proceedings of the 2nd IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, pp. 418–429 (2005)
15. Kurose, J.F., Ross, K.W.: Computer Networking – A Top-Down Approach Featuring the Internet, 5th edn. Addison-Wesley Professional, Reading (2009)
16. Shah, K., Kumar, M.: Distributed independent reinforcement learning (DIRL) approach to resource management in wireless sensor networks. In: MASS 2007: Proceedings of the 4th IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems), pp. 1–9 (2007)
17. Watkins, C.J., Dayan, P.: Q-learning. Machine Learning 8(3), 279–292 (1992)
18. Anastasi, G., Conti, M., Gregori, E., Spagoni, C., Valente, G.: Motes sensor networks in dynamic scenarios. International Journal of Ubiquitous Computing and Intelligence 1(1) (April 2007)
19. Crossbow Technology: Mica2 wireless measurement system, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf

Author Index